МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Физико-технический факультет**

**Кафедра теоретической физики и компьютерных технологий**

**КУРСОВОЙ ПРОЕКТ**

**РАЗРАБОТКА МОДУЛЯ ОБРАЗОВАТЕЛЬНОГО ПРИЛОЖЕНИЯ «АЛГОРИТМЫ СОРТИРОВКИ»**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Бабенко Ивонна Викторовна

Курс 2

Направление 09.03.02 Информационные системы и технологии

Научный руководитель

преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М. В. Сухачев

Нормоконтролер инженер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г. Д. Цой

Краснодар 2018

 **СОДЕРЖАНИЕ**

[Введение 3](#_Toc512184581)

[1 Обзор алгоритмов сортировки 4](#_Toc512184582)

[1.1 Быстрая сортировка 4](#_Toc512184583)

[1.2 Сортировка Шелла 5](#_Toc512184584)

[1.3 Сортировка слиянием 7](#_Toc512184585)

[2 Используемые языки программирования и программное обеспечение 13](#_Toc512184586)

[2.1 Язык C++ и библиотека Qt 5.5.0 13](#_Toc512184587)

[2.2 Среда разработки Qt Creator 15](#_Toc512184597)

[3 Визуализация процесса сортировки 17](#_Toc512184601)

[3.1 Создание графических сцен 17](#_Toc512184602)

[Заключение 19](#_Toc512184607)

[Список использованных источников 20](#_Toc512184608)

Реферат

Курсовой проект 22 с., 5 рис., 14 источников.

РАЗРАБОТКА МОДУЛЯ ОБРАЗОВАТЕЛЬНОГО ПРИЛОЖЕНИЯ «АЛГОРИТМЫ СОРТИРОВКИ»

Объектом разработки данного курсового проекта является приложение «алгоритмы сортировки».

Целью данного проекта является реализация образовательного модуля приложения, ориентированного на повышение эффективности и качества подготовки учащихся. Выполнение этой цели потребовало структурирования материала модуля в такой форме, чтобы обеспечить наиболее эффективное усвоение его в конкретных условиях.

В результате выполнения курсового проекта разработаны и запрограммированы три алгоритма сортировки: первый алгоритм – быстрая сортировка, второй – сортировка Шелла, третий – сортировка слиянием. Была протестирована программа и выполнено сравнение разработанных алгоритмов сортировки.

Введение

Сортировка – один из важнейших аспектов обработки данных, позволяющий ускорить и упростить этот наиважнейший в области вычислительной техники процесс. Под сортировкой в узком смысле контекста вычислительной техники понимают упорядочивание записей внутри структуры данных.

Объектом исследования является алгоритмы сортировки.

Одной из современных форм обучения является модульная система, которая сегодня становится все популярней в силу своей динамичности и высокой результативности.

Целью данного проекта является реализация образовательного модуля приложения, ориентированного на повышение эффективности и качества подготовки учащихся. Выполнение этой цели требует структурирования материала модуля в такой форме, чтобы обеспечивалось наиболее эффективное усвоение его в конкретных условиях. В результате каждый модуль имеет четко определенную структуру, в основе которой лежит структура его элементов.

Задачи - описать, разработать и запрограммировать три алгоритма сортировки: первый алгоритм – быстрая сортировка, второй – сортировка Шелла, третий – сортировка слиянием. Протестировать программу и выполнить сравнение разработанных алгоритмов сортировки.

Актуальность решения технических задач в данной области сохраняется и на сегодняшний день. Дональд Кнут в своей книге «Сортировка и поиск» утверждает, что алгоритмы сортировки занимают половину времени исполнения всех процессов обработки данных вычислительной машиной.

1. **Обзор алгоритмов сортировки**
	1. **Быстрая сортировка**

Быстрая сортировка (англ. quicksort), часто называемая qsort (по имени в стандартной библиотеке языка Си) — широко известный алгоритм сортировки, разработанный английским информатиком Чарльзом Хоаром во время его работы в МГУ в 1960 году [1].

Один из самых быстрых известных универсальных алгоритмов сортировки массивов: в среднем O(n log n) обменов при упорядочении n элементов; из-за наличия ряда недостатков на практике обычно используется с некоторыми доработками.

Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в правую элементы, чьи значения больше или равны опорному, как показано на рисунке 1.



Рисунок 1 - Массив при быстрой сортировке

Вне зависимости от того, какой элемент выбран в качестве опорного, массив будет отсортирован, но все же наиболее удачным считается ситуация, когда по обеим сторонам от опорного элемента оказывается примерно равное количество элементов. Если длина какой-то из получившихся в результате разбиения частей превышает один элемент, то для нее нужно рекурсивно выполнить упорядочивание, т. е. повторно запустить алгоритм на каждом из отрезков [2].

Таким образом, алгоритм быстрой сортировки включает в себя два основных этапа:

- разбиение массива относительно опорного элемента;

- рекурсивная сортировка каждой части массива.

Рассмотрим реализацию быстрой сортировки в рассматриваемом приложении:

void MainWindow::quickrun() {

 this->quickrun();

}

void MainWindow::quicksort(int l, int r)

{

 this->reset(); // восстанавливаем сцену

 draw(array\_quick);

 if(r==0) r=array\_quick.count()-1;

 int i = l;

 int j = r;

 int x = array\_quick[(l+r)/2];

 do {

 while (array\_quick[i]<x)i++;

 while (array\_quick[j]>x)j--;

 if(i<=j){

 if(array\_quick[i]>array\_quick[j]){

 int temp=array\_quick[i];

 array\_quick[i]=array\_quick[j];

 array\_quick[j]=temp;

 }

 draw(array\_quick);

 QTest::qWait(500);

 i++;

 j--;

 }

 }while(i<=j);

 if(i<r)quicksort(i,r);

 if(j>l)quicksort(l,j);

}

Стоит отметить, что быстрая сортировка может оказаться малоэффективной на массивах, состоящих из небольшого числа элементов, поэтому при работе с ними разумнее отказаться от данного метода. В целом алгоритм неустойчив, а также использование рекурсии в неверно составленном коде может привести к переполнению стека. Но, несмотря на эти и некоторые другие минусы, быстрая сортировка все же является одним из самых эффективных и часто используемых методов [3].

**1.2 Сортировка Шелла**

В 1959 году американский ученый Дональд Шелл опубликовал алгоритм сортировки, который впоследствии получил его имя – «Сортировка Шелла». Этот алгоритм может рассматриваться и как обобщение пузырьковой сортировки, так и сортировки вставками [4].

Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или N/2, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии N/2; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на d/2, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на d=1 проход по массиву происходит в последний раз в соответствии с рисунком 2.

Далее, на примере последовательности целых чисел, показан процесс сортировки массива методом Шелла. Для удобства и наглядности, элементы одной группы выделены одинаковым цветом [5].

Рисунок 2 – Сортировка Шелла

Первое значение, соответствующее расстоянию d равно 10/2=5. На каждом шаге оно уменьшается вдвое. Элементы, входящие в одну группу, сравниваются и если значение какого-либо элемента, стоящего левее того с которым он сравнивается, оказывается больше (сортировка по возрастанию), тогда они меняются местами. Так, элементы путем внутригрупповых перестановок постепенно становятся на свои позиции, и на последнем шаге (d=1) сортировка сводится к проходу по одной группе, включающей в себя все N элементов массива. При этом число требуемых обменов оказывается совсем небольшим [6].

Рассмотрим осуществление сортировки Шелла:

void MainWindow::shellsort()

{

 this->reset();

 d=array\_shell.size();

 d=d/2;

 while (d>0)

 {

 for (i=0; i<array\_shell.size()-d; i++)

 {

 j=i;

 while (j>=0 && array\_shell[j]>array\_shell[j+d])

 {

 count=array\_shell[j];

 array\_shell[j]=array\_shell[j+d];

 array\_shell[j+d]=count;

 draw(array\_shell);

 QTest::qWait(500);

 j--;

 }

 }

 d=d/2;

 }

}

Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем быстрая сортировка, она имеет ряд преимуществ:

- отсутствие потребности в памяти под стек;

- отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n?), что хуже, чем худшее гарантированное время для сортировки Шелла [7]

**1.3 Сортировка слиянием**

Алгоритм был изобретён Джоном фон Нейманом в 1945 году.

Сортировка слиянием (англ. merge sort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи:

- массив рекурсивно разбивается пополам, и каждая из половин делиться

до тех пор, пока размер очередного подмассива не станет равным единице;

- далее выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива. После чего из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив;

- в конце операции слияния, элементы перезаписываются из результирующего массива в исходный [8].

Существует несколько вариантов сортировки слиянием.

Слияние двух отсортированных массивов. Центральное место в алгоритме сортировки слиянием занимает слияние двух предварительно отсортированных массивов. В результате слияния двух сортированных массивов A и B создается третий массив C, который содержит все элементы A и B, также расположенные в порядке сортировки. Сначала мы рассмотрим сам процесс слияния, а потом разберемся, как он используется при сортировке.

Представьте, что у вас имеются два предварительно отсортированных массива (причем необязательно одинакового размера). Допустим, массив содержит 4 элемента, а массив B — 6. Массивы объединяются посредством слияния в массив C, изначально состоящий из 10 пустых ячеек (на рисунке 3).



а — до слияния; б — после слияния

Рисунок 3 - Слияние двух массивов

Нисходящая сортировка слиянием. Идея сортировки слиянием заключается в том, чтобы разделить массив пополам, отсортировать каждую половину, а затем воспользоваться методом merge() для слияния двух половин в один отсортированный массив. Вызов рекурсивного метода приводит к уменьшению некоторого параметра, а при каждом возвращении из метода этот параметр снова увеличивается. В mergeSort() при каждом рекурсивном вызове диапазон делится надвое, а при каждом возвращении — две половины объединяются в один больший диапазон. Когда метод mergeSort() возвращает управление после обнаружения двух массивов, содержащих по одному элементу, он объединяет их в отсортированный массив из двух элементов. Далее каждая пара полученных 2-элементных массивов объединяется в массив из 4-х элементов. Процесс продолжается с массивами постоянно увеличивающихся размеров до тех пор, пока весь массив не будет отсортирован. За сортировкой проще всего наблюдать для массивов, исходный размер которых составляет степень 2, как показано на рисунке 4 [9].



Рисунок 4 - Последовательное слияние с увеличением

размера массивов

Восходящая сортировка слиянием. У каждой рекурсивной программы имеется нерекурсивный аналог, который хотя и выполняет эквивалентные вычисления, тем не менее, он делает это по-другому.

Порядок выполнения слияний определяется рекурсивной структурой алгоритма. Однако подфайлы обрабатываются независимо и слияния могут выполнятся в различных .

Рассмотрим псевдокод, состоящий из основной подпрограммы MergeSort, которая рекурсивно разбивает и сортирует массив, и Merge, отвечающей за его слияние.

Рассмотрим реализацию сортировки слиянием**:**

void MainWindow::merge(QVector<int>& theArray, QVector<int> workspace, int lowPtr, int highPtr, int upperBound)

{

 int lowerBound = lowPtr;

 int mid = highPtr - 1;

 while (lowPtr <= mid && highPtr <= upperBound)

 if (theArray[lowPtr] < theArray[highPtr])

 workspace.push\_back(theArray[lowPtr++]);

 else

 workspace.push\_back(theArray[highPtr++]);

 while (lowPtr <= mid)

 workspace.push\_back(theArray[lowPtr++]);

 while (highPtr <= upperBound)

 workspace.push\_back(theArray[highPtr++]);

 for(int j=0;j<upperBound-lowerBound+1;j++)

 {

 theArray[lowerBound+j] = workspace[j];

 }

}

void MainWindow::recMergeSort(QVector<int>& theArray, int lowerBound, int upperBound)

{

 QVector<int> workspace;

 if (lowerBound == upperBound) return;

 else

 {

 int mid = (lowerBound + upperBound) / 2;

 recMergeSort(theArray, lowerBound, mid);

 recMergeSort(theArray, mid + 1, upperBound);

 merge(theArray, workspace, lowerBound, mid + 1, upperBound);

 draw(array\_merge);

 QTest::qWait(500);

}

}

Сортировка слиянием играет важную роль благодаря простоте и оптимальности заложенного в нее метода, который допускает возможность реализации, обладающей устойчивостью.

Недостатком сортировки слиянием является использование дополнительной памяти. Но когда работать приходиться с файлами или списками, доступ к которым осуществляется только последовательно, то очень удобно применять именно этот метод. Также, к достоинствам алгоритма стоит отнести его устойчивость и неплохую скорость работы O(n\*logn) [10].

**2** **Используемые языки программирования и программное обеспечение**

 **2.1** **Язык C++ и библиотека Qt 5.5.0**

 C++ поддерживает объектно-ориентированное программирование, обобщённое программирование через шаблон, дополнительные типы данных, встраиваемые функции, исключения и т.д... Возможности C++ включают объявления в виде выражений, преобразования типов в виде 7 функций, расширенное понятие константности, подставляемые функции, аргументы по умолчанию, переопределения, пространства имён, классы [11].

Для проекта был выбран язык программирования С++/Qt. Qt – (произносится [ kju t] ˈ ː (кьют) как «cute» или неофициально Q-T (кью- ти)) — кроссплатформенный инструментарий разработки ПО на различных языках программирования. Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q\_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий мета объектный код.

Qt 5.5.0 предоставляет программисту не только удобный набор библиотек классов, но и определённую модель разработки приложений, определённый каркас их структуры. Следование принципам и правилам «хорошего стиля программирования на C++/Qt» существенно снижает частоту таких трудно отлавливаемых ошибок в приложениях, как утечки памяти (memory leaks), необработанные исключения, незакрытые файлы или неосвобождённые дескрипторы ресурсных объектов, чем нередко страдают программы, написанные «на голом C++» без использования библиотеки Qt.

Важным преимуществом Qt 5.5.0 является хорошо продуманный, логичный и стройный набор классов, предоставляющий программисту очень высокий уровень абстракции. Благодаря этому программистам, использующим Qt, приходится писать значительно меньше кода, чем это имеет место при использовании, например, библиотеки классов MFC. Сам же код выглядит стройнее и проще, логичнее и понятнее, чем аналогичный по функциональности код MFC или код, написанный с использованием «родного» для X11 тулкита Xt. Его легче поддерживать и развивать. Qt позволяет создавать собственные плагины и размещать их непосредственно 8 в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна [12].

 Qt 5.5.0 комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы в режиме WYSIWYG. В поставке Qt есть «Qt Linguist» — графическая утилита, позволяющая упростить локализацию и перевод программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кросс-платформенную справку для разрабатываемого на основе Qt ПО. Программное обеспечение, построенное на этом инструментарии не обязательно должно содержать графический интерфейс пользователя (GUI). С помощью огромного количества всевозможных классов можно написать приложение, работающее только через интерфейс командной строки (CLI). Но в основном, конечно, Qt используется для построения графических приложений, главная особенность которых - кроссплатформенность.

Даже если программисту в данный конкретный момент не нужна кроссплатформенность для его конкретного приложения (например, планируется версия только для Windows или только для Macintosh), никто не может знать, что понадобится завтра. Планы могут поменяться, и может оказаться и нужным, и выгодным выпустить версию для другой операционной системы или другой аппаратной платформы. В случае использования Qt для этого понадобится всего лишь перекомпиляция исходного кода. В случае же использования, например, MFC или «родных» системных API понадобится много тяжёлой работы по портированию, адаптации и отладке, а то и переписыванию с нуля существующего исходного кода для другой ОС или аппаратной платформы. Qt является 9 полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Основой построения графических интерфейсов на Qt, являются такие компоненты, как виджеты. Каждый элемент в окне программы, будь то кнопка или переключатель – это отдельный виджет. Виджеты генерируют различные сигналы, которые могут вызывать слоты – специальные функции, реагирующие на различные события в окне.

 Многие компании-разработчики приложений Windows используют Qt ещё по одной причине: даже если код пишется и в обозримом будущем будет писаться только для платформы Windows и тестируется только на ней, возможность откомпилировать один и тот же исходный код на одной и той же платформе Windows двумя разными компиляторами (Microsoft Visual C+ + и GCC/Win32) гарантирует лучшее качество исходного кода и лучшую его совместимость со стандартом C++. Что немаловажно для кода, который планируется длительно поддерживать и развивать.

У библиотеки Qt существуют «привязки» ко многим популярным языкам программирования: Python, Java и другим. Эти «привязки» позволяют многим программистам писать программы с графическим интерфейсом, основанным на Qt на любимом языке, не вдаваясь во все подробности C++.

**2.2** **Среда разработки Qt Creator**

 Для работы с библиотекой Qt можно использовать следующие среды визуального программирования (Debian, Fedora, Gentoo, Mandriva, Ubuntu, Qt Creator). Для создания приложения была выбрана среда Qt Creator. Начиная с версии 4.5.0 в комплект Qt включена среда разработки «Qt Creator», которая включает в себя редактор кода, справку, графические средства «Qt Designer» и возможность отладки приложений. «Qt Creator» может использовать GCC или Microsoft VC++ в качестве компилятора и GDB в качестве отладчика. Для Windows версий библиотека 10 комплектуется компилятором, заголовочными и объектными файлами MinGW .

 Qt Creator - кроссплатформенная свободная IDE для разработки на С, С++ и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Основная задача Qt Creator — упростить разработку приложения на разных платформах. Поэтому среди возможностей, присущих любой среде разработки, есть и специфичные, такие как отладка приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов как на QML, так и на Qt Widgets. Qt Creator поддерживает системы сборки qmake, cmake, autotools. Для проектов, созданных под другими системами, может использоваться в качестве редактора исходных кодов. Есть возможность редактирования этапов сборки проекта. В Qt Сreator реализовано автодополнение, в том числе ключевых слов, введённых в стандарте C++11, подсветка кода. Также есть возможность задания стиля выравнивания, отступов и постановки скобок. Реализован ряд возможностей при работе с сигнатурами методов, а именно: автогенерация пустого тела метода после его обновления; возможность автоматически изменить сигнатуру метода в определении, если она была изменена в объявлении и наоборот; возможность автоматически поменять порядок следования аргументов.

 При навигации по коду доступно переключение между определением и объявлением метода, переход к объявлению метода, переименование метода как в отдельном проекте, так и во всех открытых. Также есть возможность вызвать справку согласно текущему контексту [13].

**3 Визуализация процесса сортировки**

**3.1 Создание графической сцены**

Одним из важнейших дидактических принципов, который лежит в основе организации и эффективности обучения и восприятия, является наглядность. Новые информационные технологии дают возможность создать динамические учебные среды, а представление процессов и явлений в движении и развитии имеет особое значение для познавательной деятельности студентов.

Отрисовка данной программы будет происходить на QGraphicsScene. Основу Graphics View Framework составляют три класса Qt Library, представленные на рисунке 5.

Рисунок 5 – Схема Graphics View Framework

Класс QGraphicsScene предоставляет поверхность для управления большим числом графических 2D элементов. Графическая сцена (QGraphicsScene) хранит внутри себя графические элементы (QGraphicsItem). Элементы упорядочиваются определенным образом (за счет этого, обеспечиваются перерисовка минимальной области сцены при изменении свойств элемента и эффективное обнаружение столкновений). [14].

Рассмотрим использование графической сцены в данном приложение:

this->setWindowTitle("Сортировка");

 this->setFixedSize(600, 600);

 glMain = new QGridLayout;

wgt = new QWidget;

wgt->setLayout(glMain);

this->setCentralWidget(wgt);

int w = 600, h = 500;

scene = new QGraphicsScene(this);

scene->setSceneRect(0,0, w, h);

gView = new QGraphicsView(scene);

gView->setFixedHeight(scene->height()+3);

gView->setFixedWidth(scene->width()+3);

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта:

 1 Был изучен ряд источников по практическому программированию. В результате чего приобретены знания необходимые для создания образовательного модуля приложения.

2 Так же в ходе решения поставленных задач было осуществлено знакомство с алгоритмами сортировок, их разработка и программирование приложения.

3 Была изучена библиотека Qt 5.5.0. В результате чего стало возможным создание приложения.

4 Были разработаны слоты сортировки быстрая (quicksort), слиянием (recMergeSort) и Шелла (shellsort). Данные слоты интегрированы в разработанное приложение, с помощью которого можно создать массив с заданным количеством элементов, отсортировать его любым из рассмотренных в курсовом проекте методом сортировки.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. EVILEG 2015-2018 - (Рус.). - URL: https://evileg.com/ru/ [3 апреля 2017].
2. Kvodo Computing Science & Discrete Match - (Рус.). - URL: http://kvodo.ru/ [3 апреля 2017].
3. Niklaus Wirt. Algoritms and Data Structures = Programs, Prentice-Hall, 1986. - (Engl.). - URL: http://corewar.co.uk/assembly/insertion.htm [3 April 2017].
4. Animations for Data Structures and Algorithms - (Engl.). - URL: http://www.csse.monash. edu.au/~dwa/Animations/index.html [3 April 2017].
5. CS 162 Program Demonstration. - (Engl.). - URL: http://web.engr.oregonstate.edu/~minoura/cs162/sort/InsertSort.htm [3 April 2017].
6. Interactive data structure visualizations. - (Engl.). - URL: http://nova.umuc.edu/ ~jarc/idsv/ [3 April 2017].
7. Data Structures and Algorithms with Object-Oriented Design Patterns in C++ - (Engl.). - URL: http://brpreiss.com/book s/opus4/html/page487.html [3 April 2017].
8. Sorting Algorithm Animations - URL: http://www.sorting-algorithms.com/ [3 April 2017].
9. Р. Лафоре Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд./ Р. Лафоре — СПб.: Питер, 2013. — 704 с.
10. The xSortLab Applet - (Engl.). - URL: http://math.hws.edu/TMCM/java/xSortLab/ index.html [3 April 2017].
11. Introduction to Algorithms / Cormen, T. H., C.E. Leiserson, R.L. Rivest, and C. Stein. - Cambridge, MA: MIT Press, 1990. - 984 с.
12. Программирование на языке С++ в среде Qt Creator / Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк, О. В. Чеснокова, А. С. Чмыхало. - ALT Linux, январь 2015. - 448 с.
13. Б.Страуструп. Язык программирования С++, 2-е изд./Пер. с англ. Часть первая. / Б.Страуструп. – Киев.: “ДиаСофт”, 1993. – 296 с.
14. Шлее М. Qt 5.3. Профессиональное программирование на С++ /М. Шлее – СПБ.: БХВ-Петербург, 2015. –928с.