Министерство Науки и Образования Украины

 **ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ**

Кафедра Информатики

Пояснительная записка

 КУРСОВАЯ РАБОТА

### ПО КУРСУ

**“Объектно-ориентированное программирование на Visual C ++”**

**на тему: "Методы сортировки. Их сравнительный анализ"**

## Выполнил: Проверил:

## *Ст. гр.* СУА-03-1 *старший преподаватель*

##  Котляров М.Н. Бритик В.И.

###

###

###  Харьков 2004

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Решение интеллектуальной задачи на компьютере

2 ПОСТРОЕНИЕ АЛГОРИТМА КОДИРОВАНИЯ НА VISUALC++

 2.1 Алгоритм решения задачи

 2.2 Описание программы “Sort”

3 Инструкции пользователя

ЗАКЛЮЧЕНИЕ

Приложение

ЛИТЕРАТУРА И ИСТОЧНИКИ

РЕФЕРАТ

#### Записка пояснительная к курсовой работе содержит: 24 стр.

Предмет исследования - современные методы разработки программ таких, как объектно-ориентированное программирование и визуальное проектирование, а также структурное и модульное программирование.

Цель курсовой работы - систематизация, углубление и активное применение знаний по системному программированию, закрепление знаний, полученных в лекционном курсе, а также на практических и лабораторных занятиях.

Метод исследования - изучение литературы, составление и отладка программ на компьютере.

Программа типа “Sort” может использоваться, как программа, предназначенная для сортировки элементов массива.

Разработан проект “Sort” полностью соответствующий условию задания и имеющий довольно удобный интерфейс.

КЛЮЧЕВЫЕ СЛОВА: SORT, Visual C++, функция, проект, сообщение, программа.

ВВЕДЕНИЕ

В настоящее время вычислительная техника проникла практически во все сферы человеческой деятельности. С помощью ЭВМ можно решать самые разные задачи. Но для того, чтобы решить поставленную задачу, необходимо указать последовательность действий, выполнение которых приведёт к требуемому результату, – составить программу. Для удобства работы с ЭВМ эта операция производится с помощью языков программирования (высокого или низкого уровня).

Один из широко используемых языков программирования - это Visual C++, который можно использовать для написания программ, работающих в операционной среде Windows. На данное время одной из самых распространенных его версий является Microsoft Visual C++, и среда программирования Microsoft Developer Studio 6.0.

Среда программирования Microsoft Developer Studio 6.0 позволяет создавать тексты программ, компилировать их, находить ошибки и оперативно их исправлять; компоновать программы из отдельных частей, включая стандартные модули, отлаживать и выполнять отлаженную программу.

Используя перечисленные возможности, можно создавать различные прикладные программы, например, такие, как программа, написанная при выполнении данной курсовой работы.

***1*** *Решение интеллектуальной задачи на компьютере*

В данном курсовом проекте необходимо разработать программу типа "Sort", с помощью которой можно производить сортировку массива различными методами. В частности в данном курсовом проекте используются следующие методы: “Обменная сортировка с разделением (quicksort)”, “Метод Шелла” и “Метод прямого обмена (Пузырька)”. Программа должна иметь удобный интерфейс.

***2*** *ПОСТРОЕНИЕ АЛГОРИТМА КОДИРОВАНИЯ НА VISUAL C++*

Среда Visual C++ - это сложный механизм, обеспечивающий высокоэффективную работу программиста. Создание прикладных программ, или приложений выполняется в интегрированной среде разработки IDE (Integrated Development Environment). IDE служит для организации взаимодействия с программистом и включает ряд окон, содержащих различные управляющие элементы. С помощью средств интегрированной среды разработчик может проектировать интерфейсную часть приложения, а также писать программный код и связывать его с управляющими элементами. При этом вся работа по созданию приложения, включая отладку, происходит в IDE.

Интегрированная среда разработки Visual C++ представляет собой многооконную систему. Вид интегрированной среды разработки (интерфейс) может различаться в зависимости от настроек. Кроме стандартных окон, на экране могут присутствовать и другие окна, отображаемые при вызове соответствующих средств, например, Image Editor (Редактор изображений). Окна Visual C++ (но не главное) можно перемещать, убирать с экрана, а также изменять их размеры.

Несмотря на наличие многих окон, Visual C++ является одно-документной средой, т.е. позволяет одновременно работать только с одним приложением (проектом приложения). Название проекта приложения выводится в строке заголовка главного окна в верхней части экрана.

Если свернуть главное окно, то происходит минимизация всего интерфейса Visual C++ и, соответственно, всех открытых окон. При закрытии главного окна работа с Visual C++ прекращается.

Самой последней и наиболее усовершенствованной версией стал Microsoft Visual C++ 6.0. Visual C++ 6.0, вобрав в себя всё самое лучшее от предыдущих версий, предоставляет ряд новых возможностей. Так, например, стал более удобным и современным интерфейс среды программирования, создаваемые Visual C++программы учитывают архитектуру современных процессоров, существенно расширены возможности отладчика.

Visual C++ 6.0 может работать в среде операционных систем от Windows 95 до Windows 2000. Особенных требований к компьютеру система не предъявляет, за исключением того, что процессор должен быть типа Pentium, оперативной памяти - не менее 32 Мбайт и достаточное количество свободной дисковой памяти (порядка 200 Мбайт).

***2.1 Алгоритм решения задачи***

Основными операциями, выполняемыми над массивами, являются упорядочение (сортировка) записей и поиск в массиве  записи по заданному условию( по ключу ). Сортировка является операцией расстановки записей массива в определенном порядке в соответствии с некоторым критерием упорядочения. Сортировка осуществляется в соответствии со значением ключей всех записей (напр., упорядочение фамилий по алфавиту или чисел по возрастанию ). Существует достаточно много методов     сортировки, принципиально отличающихся друг от друга. Если массив целиком помещается в оперативной памяти ЭВМ, то его упорядочение называют внутренним. Если для хранения упорядочиваемых данных используются внешнее запоминающее устройство, то такое упорядочение называют внешним. Критериями оценки методов сортировки являются:

 С -    количество операций сравнения пар ключей,

   Р -    число перестановок элементов ,

   S -    резерв памяти.

Среднее количество операций    сравнения зависит от    метода сортировки и при рациональном выборе метода достигает некоторого минимума, зависящего от n - размера массива (размер массива - количество содержащихся в нём записей). Методы внутренней сортировки можно разделить на две группы:

* методы, не требующие резерва памяти;
* методы, требующие резерва памяти.

К первой группе относятся такие методы, как метод выборки, "пузырька", вставки, Шелла. Ко второй группе относятся метод квадратичной выборки,    метод слияния и другие. Простые методы сортировки (выбором,     обменом, вставкой) требуют приблизительно n\*\*2 сравнений. Более сложные алгоритмы обычно обеспечивают получение результата за n\*log2(n) сравнений в среднем: сортировка методом Шелла, слиянием, "быстрая сортировка". Однако оптимальной в любом случае сортировки не существует, так как их эффективность существенно зависит от типа ключей в массиве и их предварительной упорядоченности.
Рассмотрим алгоритмы наиболее распространенных методов внутренней сортировки ( упорядочение выполняется по возрастанию значений ключа ).

* Метод "Пузырька".

При использовании этого способа требуется самое большее (n-1) проходов. В течение первого прохода массива сравниваются ключи К1 и К2 первой и второй записей, и, если порядок между ними нарушен,    то записи R1 и R2 меняются местами. Затем этот процесс повторяется для записей R2 и R3, R3 и R4 и т.д. Данный метод заставляет двигаться, или "всплывать", записи с малыми ключами.    После первого прохода запись с наибольшим    ключом будет находиться на n - й позиции массива. При каждом последующем проходе записи со следующем наибольшим ключом будут располагаться в позициях n-1,    n-2, ... , 2 соответственно, в результате чего будет сформирована отсортированная таблица. После каждого прохода через массив может быть сделана проверка, были ли совершены перестановки в течение данного прохода. Если перестановок не было, то это означает, что массив уже отсортирован и дальнейших проходов не требуется. Кроме того, можно запоминать индекс последней перестановки. Это позволит уменьшить на следующем шаге просматриваемый массив.
Характеристики сортировки методом "пузырька" в худшем случае составляют n(n-1)/2 сравнений и n(n-1)/2 перестановок (худшим считается случай,когда элементы наиболее удалены от своих конечных позиций). Среднее число сравнений и перестановок имеет порядок n\*\*2 . Сортировка пузырьковым методом использует метод обменной  сортировки. Она основана на выполнении в цикле операций сравнения   и при необходимости обмена соседних элементов. Ее название происходит из-за подобия процессу движения пузырьков в резервуаре с  водой когда каждый пузырек находит свой собственный уровень.

Сортировку пузырьковым методом можно в    некоторой степени   улучшить и тем самым немного улучшить ее временные характеристики. Можно, например, заметить, что сортировка пузырьковым методом    обладает одной особенностью: расположенный не на своем месте в  конце массива элемент (например, элемент "а" в массиве "dcab")  достигает своего места за один проход, а элемент, расположенный в начале массива (например, элемент "d"), очень медленно достигает своего места.     Необязательно все просмотры делать в одном направлении. Вместо этого всякий последующий просмотр можно делать в противоположном направлении.    В этом случае сильно удаленные от своего места элементы будут быстро перемещаться в соответствующее место. Хотя эта сортировка является улучшением пузырьковым методом,  ее нельзя рекомендовать для использования, поскольку время выполнения по-прежнему зависит квадратично от числа элементов. Число сравнений не изменяется, а число обменов уменьшается лишь на незначительную величину.

* Метод Шелла.

Общий метод, который использует сортировку вставкой, применяет принцип уменьшения расстояния между сравниваемыми элементами. Сначала сортируются все элементы, которые смещены  друг от друга на три позиции. Затем сортируются все элементы, которые смещены на две позиции. И, наконец, упорядочиваются все соседние элементы.

При поверхностном взгляде на алгоритм нельзя сказать, что он  дает хороший результат и даже то, что в результате получится отсортированный массив.    Однако, он дает и то и другое. Эффективность этого алгоритма объясняется тем, что при каждом проходе используется относительно небольшое число элементов или элементы  массива уже находятся в относительном порядке, а упорядоченность   увеличивается при каждом новом просмотре данных.

Расстояния между сравниваемыми элементами могут изменяться  по-разному. Обязательным является лишь то, что последний шаг должен равняться единице. Например, хорошие результаты дает последовательность шагов 9, 5, 3, 2, 1, которая использована в данной курсовой работе. Следует избегать последовательностей степени  двойки, которые, как показывают сложные математические выкладки, снижают эффективность алгоритма сортировки. /Однако, при использовании таких последовательностей шагов между сравниваемыми элементами эта сортировка будет по-прежнему работать правильно. Слегка измененные версии сортировки Шелла используют специальные управляющие элементы, которые не являются в действительности частью той информации, которая  должна сортироваться. Управляющие элементы имеют граничные для   массива данных значения, т.е. наименьшее и наибольшее значения. В  этом случае не обязательно выполнять проверку на граничные значения. Однако, применение таких управляющих элементов требует специальных знаний о той информации, которая сортируется, и это снижает универсальность процедуры сортировки. Анализ сортировки Шелл требует решения некоторых сложных   математических задач. Время  выполнения сортировки    Шелла пропорционально n\*\*1.2. Эта зависимость значительно лучше квадратичной зависимости, которой подчиняются рассмотренные ранее алгоритмы сортировки. Однако, прежде   чем вы решите использовать сортировку Шелла, следует иметь в виду, что быстрая сортировка дает даже еще лучшие результаты.

В рассмотренных алгоритмах сортировки запись перемещается каждый раз только на одну позицию. При этом среднее время работы будет в лучшем случае пропорционально n. Методом, существенно превосходящим простые вставки, при котором записи перемещаются большими скачками, является метод Шелла (сортировка с убывающим шагом). Метод состоит в том, что упорядоченный массив разделяется на группы элементов, каждая из которых упорядочивается методом вставки. В процессе упорядочения размеры таких групп увеличиваются до тех пор, пока все элементы массива не войдут в упорядоченную группу. Выбор очередной упорядочиваемой группы и ее расположение внутри массива производятся так, чтобы можно было использовать предшествующую упорядоченность. Группой массива называют последовательность элементов, номера которых образуют арифметическую прогрессию с разностью h (h называют шагом группы). В начале процесса упорядочения выбирается первый шаг группы h1, который зависит от размера таблицы. Шелл предложил брать

    h1=[n/2], а hi=h((i-1)/2).

В более поздних работах Хиббард показал, что для ускорения процесса целесообразно определить шаг h1 по формуле:

h1=2\*\*k+1 , где 2\*\*k < n <= 2\*\*(k+1).

После выбора h1 методом вставки упорядочиваются группы, содержащие элементы с номерами позиций

i, i+h1, i+2\*h1, ..., i+mi\*h1.

При этом i=1,2,...,h1; m[i] - наибольшее целое, удовлетворяющее неравенству i+m[i]\*hi <= n.

Затем выбирается шаг h2<h1 и упорядочиваются группы, содержащие элементы с номерами позиций i, i+h2, ..., i+m[i]\*h2. Эта процедура со все уменьшающимися шагами продолжается до тех пор, пока очередной шаг h[l] станет равным единице (h1>h2>...>hl). Этот последний этап представляет собой упорядочение всего массива методом вставки. Но так как исходный массив упорядочивался отдельными группами с последовательным объединением этих групп, то общее количество сравнений значительно меньше,     чем n /4, требуемое при методе вставки. Число операций сравнения пропорционально n\*(log2(n))\*\*2 .

* Обменная сортировка с разделением (Quicksort).

Данный метод является одним из лучших методов внутренней сортировки и весьма эффективен для больших массивов. Целью каждого шага в данном методе является помещение очередной рассматриваемой записи на конечную позицию внутри массива. Если поступать таким способом, то все записи, предшествующие данной, будут иметь меньший ключ, в то время как все последующие - больший. При использовании такого метода массив всегда делится на два подмассива. Анологичный процесс может быть применен к каждому из этих подмассивов и повторяться до тех пор, пока все записи не будут установлены на их конечные позиции. Используются два индекса i и j с начальными значениями границ массива. Сравниваются ключи K[i] и K[j], и если перестановка не требуется, то j уменьшается на 1, и процесс повторяется.     В том    случае, когда K[i]>=K[j], записи R[i] и R[j] меняются местами. Затем этот процесс повторяется с i, увеличенным на 1, и фиксированным j до тех пор, пока не возникает другая перестановка. В этот момент j снова будет уменьшено на 1, а i останется фиксированным, и т.д. Процесс выполняется до тех пор, пока i<j.

Каждый раз массив разбивается на два подмассива, один из которых обрабатывается, в то время как границы второго запоминаются с тем,     чтобы он был обработан позже. В этих целях может быть использован стек, представляющий собой матрицу из двух столбцов. В первом столбце хранятся нижние границы разделяемых подмассивов, во втором - верхние границы. Всякий раз один из полученных в результате разделения подмассивов подвергается дальнейшему разделению, а границы другого помещаются в свободную строку стека (обычно в стек помещаются границы большего по длине массива, поскольку это уменьшает требуемый размер стека). Как только завершается процесс разделения текущего подмассива, т.е. ее длина станет меньше трех, для разделения берется подмассив, границы которого были занесены в стек последними. Строка стека, в которой хранились эти границы, освобождается. Процесс упорядочения завершается, когда стек полностью освобождается.

Разделение следует применять для подмассивов,     длина которых больше 9, а короткие подмассивы упорядочивать методом вставки.

Стек занимает мало места. Например, стек из 20 строк позволяет упорядочить массив, содержащую до 10\*\*7 записей. Кроме того, в современных языках программирования    при работе рекурсивных программ занесение и извлечение из стека выполняется автоматически, поэтому рекомендуется использовать именно этот механизм. Среднее число сравнений для данного алгоритма составляет n\*log(n) где n - число записей в сортируемом массиве, m - размер подмассива, сортируемой методом вставки.

Наихудшей ситуацией при использовании рассмотренного алгоритма является случай, когда массив уже отсортирован. При этом число сравнений порядка n, т.е. алгоритм не лучше сортировки выборкой.

***2.2 Описание программы “Sort”***

Данный проект создавался с помощью AppWizard – генератором кода, создающим рабочую заготовку Windows-приложения с теми компонентами, именами классов и исходными файлами, что были указаны через диалоговые окна. В частности: в закладке Project выбираем - MFC AppWizard (exe). Затем нужно пройти серию экранов AppWizard:

Step 1: выбираем “Single document”

Step 2: оставляем без изменения

Step 3: без изменения

Step 4: оставляем только флажки “3D controls”, “Docking ToolBar”

Step 5: устанавливаем “As a statically linked library”

Step 6: отображает информацию о созданных классах

После этого AppWizard сгенерирует код для поддержки функциональных возможностей программы на базе библиотеки MFC, т.е. создаст каркас приложения. Рассмотрим некоторые элементы программы, созданные на данном этапе:

Класс **CSortApp.** Объект класса CSortApp представляет программу. В программе определяется единственный глобальный объект класса CSortApp – theApp. Базовый класс CWinApp определяет основные характеристики объекта theApp.

Класс **CSortView.** Объект класса CSortView представляет основное окно программы. Когда конструктор вызывает функцию-член Create базового класса CFrameWnd, Windows создаёт действительную оконную структуру, а каркас приложения связывает её с C++-объектом. Функции ShowWindow и UpdateWindow, являющиеся также функциями-членами базового класса, вызываются для вывода окна на экран.

 При запуске проекта операционная система вызывает в программе функцию WinMain, а та ищет глобально сконструированный объект класса, производного от CWinApp. В любом приложении, в том числе и в “Sort”, обязательно должна присутствовать эта функция, на которую возлагается ряд специфических задач. И самая важная – создание основного окна программы, с которым должен быть связан код, способный обрабатывать сообщения, передаваемые операционной системой этому окну. В нашем случае при программировании на Microsoft Visual C++ 6.0, с библиотекой классов Microsoft Foundation Class (MFC) Library 6.0, эта функция скрыта внутри каркаса приложения и нет необходимости в её написании, но необходимо понимать, что именно с помощью неё осуществляется связь между операционной системой и программой.

Библиотека MFC прямо поддерживает около 140 функций, обрабатывающих Windows-сообщения. Кроме того, можно определять свои собственные сообщения, связанные с обработчиками команд меню, элементов управления и т.д. В программе “Sort” используется более 40 функций, методов и сообщений Windows. Ниже они перечислены в порядке их появления в программе с кратким описанием:

Format – преобразует типы переменных;

InvalidateRect и Invalidate – обновляют рабочую область и генерируют сообщение WM\_PAINT;

DestroyWindow – разрушает окно;

PostQuitMessage – посылает окну сообщение WM\_DESTROY;

ShowWindow – отображает или скрывает окно;

UpdateWindow – заставляет окно перерисовать своё содержимое;

TextOut – вывод текста на экран;

После вызова функции UpdateWindow, окно окончательно выведено на экран. Теперь программа должна подготовить себя для получения информации от пользователя через клавиатуру и мышь. Windows поддерживает “очередь сообщений” (message queue) для каждой программы, работающей в данный момент в системе Windows. Когда происходит ввод информации, Windows преобразует её в “сообщение”, которое помещается в очередь сообщений программы. Каждое получаемое окном сообщение идентифицируется номером, который содержится в параметре message оконной процедуры. В заголовочных файлах Windows определены идентификаторы, начинающиеся с префикса WM (“window message”) для каждого типа сообщений. Ниже приведены все сообщения используемые в курсовом проекте:

Сообщение WM\_CREATE – это первое сообщение, которое Windows посылает объекту View. Оно передаётся, когда каркас приложения вызывает оконную функцию Create, т.е. в тот момент, когда создание окна ещё не закончено и его не видно на экране. Следовательно, обработчик OnCreate пока не может обращаться к Windows-функциям, доступным только после отображения окна. Такие функции можно вызвать из замещённой функции OnInitialUpdate.

Функция-членOnDraw().Это виртуальная функция-член класса CView; каркас приложений вызывает её всякий раз, когда приходит сообщение о том, что нужно перерисовать окно отображения. А такая необходимость возникает при масштабировании окна или при появлении ранее скрытой его части, или при изменении программой данных, связанных с этим окном. В первых двух случаях каркас приложения вызывает OnDraw, но если какая-то функция в программе изменяет данные, она должна уведомить об этом Windows, вызвав наследуемую функцию-член Invalidate (или InvalidateRect) для данной области отображения. Вызов Invalidate приводит впоследствии к автоматическому вызову OnDraw.

Windows не разрешает прямой доступ к видеооборудованию, обращение к нему проходит через так называемый контекст устройства (device context), сопоставленный с конкретным окном. В библиотеке MFC контекст устройства – это С++-объект класса CDC, передаваемый функции OnDraw (по указателю) как параметр. Получив указатель на контекст устройства, можно вызывать множество функций-членов CDC, которые и выполняют всю работу по рисованию.

В данном курсовом проекте при вызове функции OnDraw происходит вывод исходного и отсортированного массива на экран, а также информации о количестве перестановок произведённых во время сортировки.

Когда пользователь выбирает пункт меню, Windows посылает программе сообщение WM\_COMMAND, содержащее идентификатор этого пункта меню в младшем слове параметра сообщения. Ниже рассмотрены идентификаторы, соответствующее пунктам меню программы:

ID\_QUIK – это идентификатор пункта “ Обменная сортировка с разделением (quicksort)” в меню. Выбор этого пункта приводит к сортировке массива данным методом.

ID\_SHELL – это идентификатор пункта “Метод Шелла” в меню. Выбор этого пункта приводит к сортировке массива методом Шелла.

ID\_PUZIROK – этому идентификатору в меню соответствует пункт “Метод прямого обмена (Пузырька)”. Выбор этого пункта приводит к сортировке массива методом “Пузырька”.

Выбор пункта меню “О программе…”, которому соответствует идентификатор ID\_APP\_ABOUT, выведет модальное окно диалога, в котором содержится краткая информация о разработчике и программе.

ID\_APP\_EXIT – этому идентификатору в меню соответствует пункт “Выход”. При выборе этого пункта происходит вызов функции OnDestroy, что приводит к разрушению окна и завершению работы с программой.

***3*** *ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ*

Запуск программы осуществляется при открытии файла Sort.exe, который находится на дискете. При этом на экране появиться окно, в левой верхней части которого будет видна надпись “Методы сортировки” – это имя программы. Ниже располагается меню, с помощью которого можно выполнить различные действия с данным приложением. При нажатии на пункте меню “Файл”, выпадет, так называемое, всплывающее меню, в котором находится пункт “Выход”. При выборе этого пункта программа закрывается.

Следующий пункт главного меню – это “Сортировка”, подменю которого содержит пункты “Обменная сортировка с разделением (quicksort)”, “Метод Шелла” и “Метод прямого обмена (Пузырька)”. Выбор первого пункта позволяет произвести сортировку массива методом “Обменной сортировки с разделением”. Нажатие на пункте меню “Метод Шелла” приводит к сортировке массива методом Шелла. И выбор последнего подпункта меню сортирует массив методом “Пузырька”.

Последним пунктом меню является “Помощь”. Если выбрать этот пункт, то в подменю можно увидеть пункт: “О программе”, который содержит информацию о разработчике и о самой программе.

Под меню располагается панель инструментов, которая дублирует все пункты основного меню. Ещё ниже расположена клиентская область, в которой происходит весь вывод информации.

Системные требования: Pentium 133, 16 MB RAM, Windows 95/98/2000 NT/XP.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данного курсового проекта были разработана программа на языке высокого уровня Visual C++. А также изучены возможности данного языка.

Систематизированы и закреплены практические навыки использования ЭВМ, программного обеспечения, существующих средств обслуживания системных программистов, а также теоретические знания по основным разделам курса "Объектно-ориентированного программирования". Основное внимание уделено изучению современных методов защиты информации, способов проектирования приложений, объектно-ориентированному и системному программированию.

При выполнении курсового проекта произведено знакомство с реферативными журналами и другими информационными источниками по объектно-ориентированному и системному программированию с целью анализа состояния решаемой задачи.

Получены практические навыки работы в средеVisual C++.

ПРИЛОЖЕНИЕ

#include "stdafx.h"

#include "Sort.h"

#include "SortDoc.h"

#include "SortView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

//объявление глобальных переменных

int mas[20]={30,5,17,8,1,14,12,3,77,2,45,89,33,21,6}, mas2[20], kol=15, count=0;

CString str;

bool sort=false;

int metod=0;

//1- quicksort

//2- shell

//3- пузырька

/////////////////////////////////////////////////////////////////////////////

// CSortView

IMPLEMENT\_DYNCREATE(CSortView, CView)

BEGIN\_MESSAGE\_MAP(CSortView, CView)

 //{{AFX\_MSG\_MAP(CSortView)

 ON\_COMMAND(ID\_QUICK, OnQuick)

 ON\_COMMAND(ID\_PUZIROK, OnPuzirok)

 ON\_COMMAND(ID\_SHELL, OnShell)

 //}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CSortView construction/destruction

CSortView::CSortView()

{

 // TODO: add construction code here

}

CSortView::~CSortView()

{

}

BOOL CSortView::PreCreateWindow(CREATESTRUCT& cs)

{

 // TODO: Modify the Window class or styles here by modifying

 // the CREATESTRUCT cs

 return CView::PreCreateWindow(cs);

}

/////////////////////////////////////////////////////////////////////////////

// CSortView drawing

//функция вывода данных на экран

void CSortView::OnDraw(CDC\* pDC)

{

 CSortDoc\* pDoc = GetDocument();

 ASSERT\_VALID(pDoc);

 // TODO: add draw code for native data here

 int i;

 //выводим исходный массив на экран

 for(i=0;i<kol;i++)

 {

 str.Format("%d,",mas[i]);//формирование строки

 pDC->TextOut(10+i\*20,10,str);//вывод на экран

 }

 //если был выбран какой-нибудь метод сортировки

 if(sort)

 {

 if(metod==1)//если выбран Quicksort

 pDC->TextOut(10,40,"Обменная сортировка с разделением (quicksort)");//вывод строки на экран

 if(metod==2)//если выбран Shell

 pDC->TextOut(10,40,"Метод Шелла");//вывод строки на экран

 if(metod==3)//если выбран Bubble

 pDC->TextOut(10,40,"Метод прямого обмена (Пузырька)");//вывод строки на экран

 //выводим отсортированный массив

 for(i=0;i<kol;i++)

 {

 str.Format("%d,",mas2[i]);//формирование строки

 pDC->TextOut(10+i\*20,80,str);//вывод строки на экран

 }

 str.Format("Количество перестановок в нашем случае: %d",count);//формирование строки

 pDC->TextOut(10,110,str);//вывод строки на экран

 if(metod==3)//если был выбран метод "Пузырька"

 {

 str.Format("Максимальное количество перестановок для массива из %d элементов методом 'Пузырька': %d",kol, kol\*(kol-1)/2);//формирование строки

 pDC->TextOut(10,140,str);//вывод строки на экран

 }

 }

}

/////////////////////////////////////////////////////////////////////////////

// CSortView diagnostics

#ifdef \_DEBUG

void CSortView::AssertValid() const

{

 CView::AssertValid();

}

void CSortView::Dump(CDumpContext& dc) const

{

 CView::Dump(dc);

}

CSortDoc\* CSortView::GetDocument() // non-debug version is inline

{

 ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CSortDoc)));

 return (CSortDoc\*)m\_pDocument;

}

#endif //\_DEBUG

/////////////////////////////////////////////////////////////////////////////

// CSortView message handlers

//при выборе сортировки Quicksort

void CSortView::OnQuick()

{

 //объявление локальных переменных

 sort=true;

 metod=1;

 int i;

 count=0;

 for(i=0;i<kol;i++)

 {

 mas2[i]=mas[i];

 }

 quicksort(0, kol-1);//вызов функции quicksort

 Invalidate(true);//перерисовка содержимого окна

}

//при выборе сортировки Shell

void CSortView::OnShell()

{

 //объявление локальных переменных

 sort=true;

 metod=2;

 int ii,t=5,i,j, k, s, m, h[6], x;

 count=0;

 for(ii=0;ii<kol;ii++)

 {

 mas2[ii]=mas[ii];

 }

 h[1]=9; h[2]=5; h[3]=3; h[4]=2; h[5]=1;

////////////////////////////////////////////

//АЛГОРИТМ

 for(m=1;m<=t;m++)

 {

 k=h[m];

 s=-k;

 for(i=k+1; i<=kol;i++)

 {

 x=mas2[i];

 j=i-k;

 while (x<mas2[j] && j<kol)

 {

 mas2[j+k]=mas2[j];

 j=j-k;

 }

 mas2[j+k]=x;

 count++;

 }

 }

 x=mas2[0];

 mas2[0]=mas2[1];

 mas2[1]=x;

////////////////////////////////////////////

 Invalidate(true);//перерисовка содержимого окна

}

//при выборе сортировки Bubble

void CSortView::OnPuzirok()

{

 //объявление локальных переменных

 int dop;

 bool end;

 count=0;

 sort=true;

 metod=3;

 int i, j;

 for(i=0;i<kol;i++)

 {

 mas2[i]=mas[i];

 }

////////////////////////////////////////////

//АЛГОРИТМ

 for(i=0;i<kol;i++)

 {

 end=true;

 for(j=i+1;j<kol;j++)

 {

 if(mas2[i]>mas2[j])

 {

 dop=mas2[i];

 mas2[i]=mas2[j];

 mas2[j]=dop;

 end=false;

 count++;

 }

 }

 if(end==true) break;

 }

/////////////////////////////////////////////

 Invalidate(true);//перерисовка содержимого окна

}

//функция быстрого поиска

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

{

 int i, j;

 i=l;j=r;

 {

 part(l, r, i, j);

 if(i<r)quicksort(i, r);// переход к сортировке левой части

 if(j>l)quicksort(l, j);// переход к сортировке правой части

 }

}

//функция поиска по частям

void CSortView::part(int l, int r, int &i, int &j)

{

 int x, dop;

 i=l;

 j=r;

 x=(l+r)/2;

 do

 {

 while(mas2[i]<mas2[x])

 i++;

 while(mas2[j]>mas2[x])

 j--;

 if(i<=j)

 {

 dop=mas2[i];

 mas2[i]=mas2[j];

 mas2[j]=dop;

 i++;j--;count++;

 }

 }

 while(i<j);

}

**Литература**

1. Петзольд Ч. Программирование под Windows 95. В двух книгах: BHV – Санкт - Петербург, 1997, silt.

2. Ричард С.Линкер, Том Арчер. Программирование для Windows 98. Библия разработчика. “Диалектика ” – Москва, 1999.-864 с.: ил.- Парал. тит. англ. Уч.пос.

3. Джесс Либерти. С++ за 21 день. ”Вильямс” - Москва, 2000.-816 с.: ил. - Парал.тит. англ.

4. Дэвид Дж. Круглински. Основы С++. “Русская редакция” – Москва, 1997.- 696 с.: ил.

5. Кэйт Грегори. Использование Visual C++. “Вильямс” – Москва, 1999.-864 с.: ил.. - Парал.тит. англ., уч. пос.

7. Конспект лекций.