Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Программное средство «CLEANER»**

|  |  |  |
| --- | --- | --- |
| Студент |  | М.Г. Гладкий |
| Руководитель |  | Д.В. Деменковец |

Минск 2020

Содержание

[Введение 5](#_Toc42242688)

[1. Анализ предметной области 7](#_Toc42242689)

[1.1. Обзор аналогов 7](#_Toc42242690)

[1.2. Формирование требований 10](#_Toc42242691)

[2. Разработка программного средства 11](#_Toc42242692)

[2.1. Структура программы 11](#_Toc42242693)

[2.2. Серверная часть 11](#_Toc42242694)

[2.3. Клиентская часть 12](#_Toc42242695)

[2.4. Удалённый доступ 13](#_Toc42242696)

[2.5. Передаваемые сообщения 16](#_Toc42242697)

[2.6. Графическое взаимодействие 18](#_Toc42242698)

[2.7. Системные сообщения 20](#_Toc42242699)

[3. Руководство пользователя 22](#_Toc42242700)

[3.1. Правила игры 22](#_Toc42242701)

[3.2. Работа с программой 22](#_Toc42242702)

[Заключение 25](#_Toc42242703)

[Список использованных источников 26](#_Toc42242704)

[Приложение А 27](#_Toc42242705)

Введение

На сегодняшний день можно с уверенностью сказать, что компьютеры проникли во все сферы деятельности человека, начиная с начального образования и заканчивая изучением новейших технологий, изучения новых видов материи, неизвестных пока человечеству. Благодаря разнообразию программного и аппаратного обеспечения сегодня возможно использование всех потенциальных возможностей компьютерных технологий. Это позволяет хранить огромное количество информации, занимая при этом минимальное место. Также компьютерные технологии позволяют быстро эту информацию обрабатывать и держать ее в защищенном виде. На разработку новых проектов затрачивается гораздо меньше времени, ибо не надо тратить массу времени на вычислительные процессы и можно полностью посвятить время самому процессу. Большую роль компьютерные технологии играют в медицине, создаются различные виртуальные модели развития заболеваний, создаются огромные базы информации, на основании которых изобретаются новые препараты для лечения.

Однако не всё так просто, как этого хотелось бы. Во время работы персональный компьютер обрабатывает огромное количество различной информации, из-за чего со временем начинает работать медленнее. Всему этому способствуют временные файлы, которые операционная система создаёт, однако в последующем не использует, неправильно удалённые программы, остаточные данные которых продолжают храниться на физическом носителе. Для обычного пользователя это может стать большой проблемой, ведь новые комплектующие могут обойтись недёшево, а как решить данную проблему без замены оборудования он не знает. Для этого необходимо проводить запланированную очистку и оптимизацию операционной системы. При недостатке свободного пространства система начинает экономить место, снижая производительность. Несмотря на то, что Windows и программы все время совершенствуются и улучшаются, свободное место на системном диске уменьшается изо дня в день, от обновления к обновлению. В результате чего требуется очистка дисков, которая удалит все ненужные файлы и папки, высвободив тем самым полезное дисковое пространство.

На помощь приходят утилиты и скрипты, занимающиеся этим нудным делом, а именно поиском мусора и его уничтожением. Существует множество таких помощников с разными функциями, возможностями и степенью доступности. Одной из самых популярных программ в этой области является CCleaner, выпущенная в 2003-м году компанией Piriform. Это утилита используется для очистки потенциально нежелательных файлов и неиспользуемых записей в реестре операционной системы Windows. Изначально утилита была разработана только для использования в среде Microsoft Windows, однако в 2012-м году была выпущена версия и для операционной системы компьютеров Apple MacOS.

Программа имеет простой и понятный пользовательский интерфейс, который позволяет рядовому потребителю выполнить очистку своего компьютера в считанные минуты. У средства также есть профессиональная версия за дополнительную плату, которая имеет больший функционал и позволяет владеть большим количеством информации о своём компьютере.

1. Анализ предметной области

1.1. Обзор аналогов

На сегодняшний день существует множество аналогов программного средства Cleaner и большинство из них реализовано для устройств на платформе Microsoft Windows.

Одним из наиболее популярных является CCleaner от компании Piriform. В данном приложении реализовано большое количество функций, таких как: проверка здоровья персонального компьютера, включающая улучшение производительности, улучшение защиты, защиту конфиденциальных данных и освобождение места на диске, стандартная очистка, очистка неиспользуемых записей реестра, поиск дублей, стирание дисков и многое другое. Главное меню утилиты на рисунке 1.1.

Достоинства:

- минималистичный и понятный интерфейс;

- большое количество возможностей;

- малый вес приложения.

Недостатки:

- необходимость покупки профессиональной версии;

- сбор данных о пользователе.

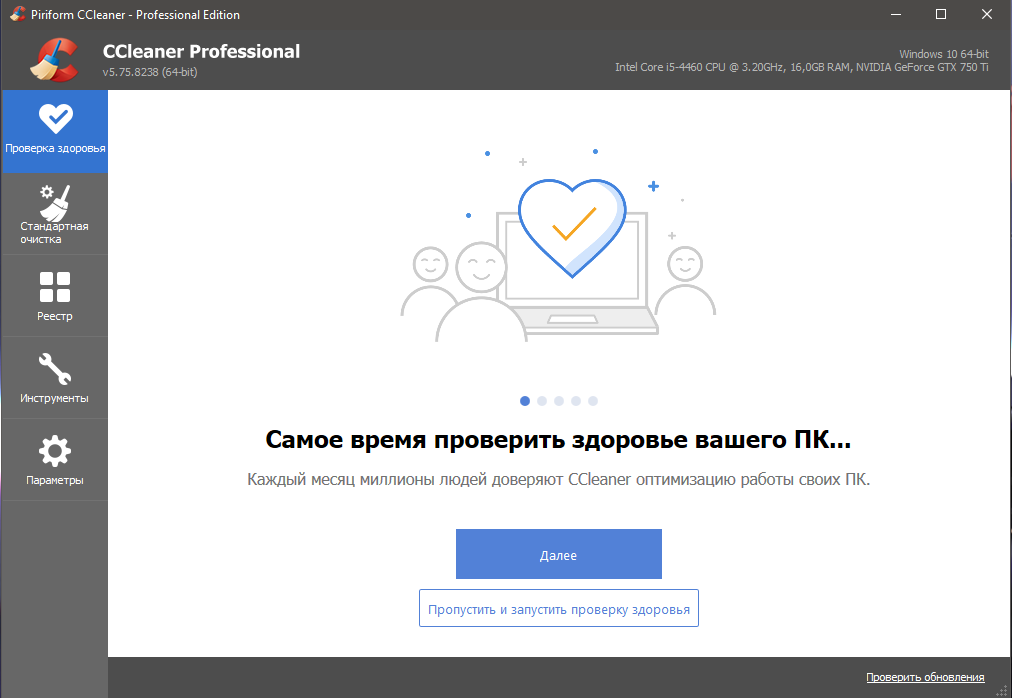


Рисунок 1.1 – Главное меню CCleaner.

На просторах сети интернет можно найти менее популярные утилиты. Одна из них – IObit Uninstaller. Основная его функция — деинсталляция любых установленных на компьютер программ. Данная функция практически повторяет функционал стандартного деинсталлятора Windows, за исключением нескольких моментов. Во-первых, IObit Uninstaller способен удалять с диска файлы и папки, оставшиеся в каталоге установки приложения. Во-вторых, утилита предлагает создание точки восстановления системы при каждой деинсталляции на случай, если пользователь захочет вернуть удаленную программу либо файлы, которые были удалены вместе с ней. Главное окно приложения представлено на рисунке 1.2.

Достоинства:

- удобное расположение элементов в приложении;

- поддержка старых операционных систем.

Недостатки:

- ограниченность функционала удалением программ;

- отсутствие кроссплатформенности.

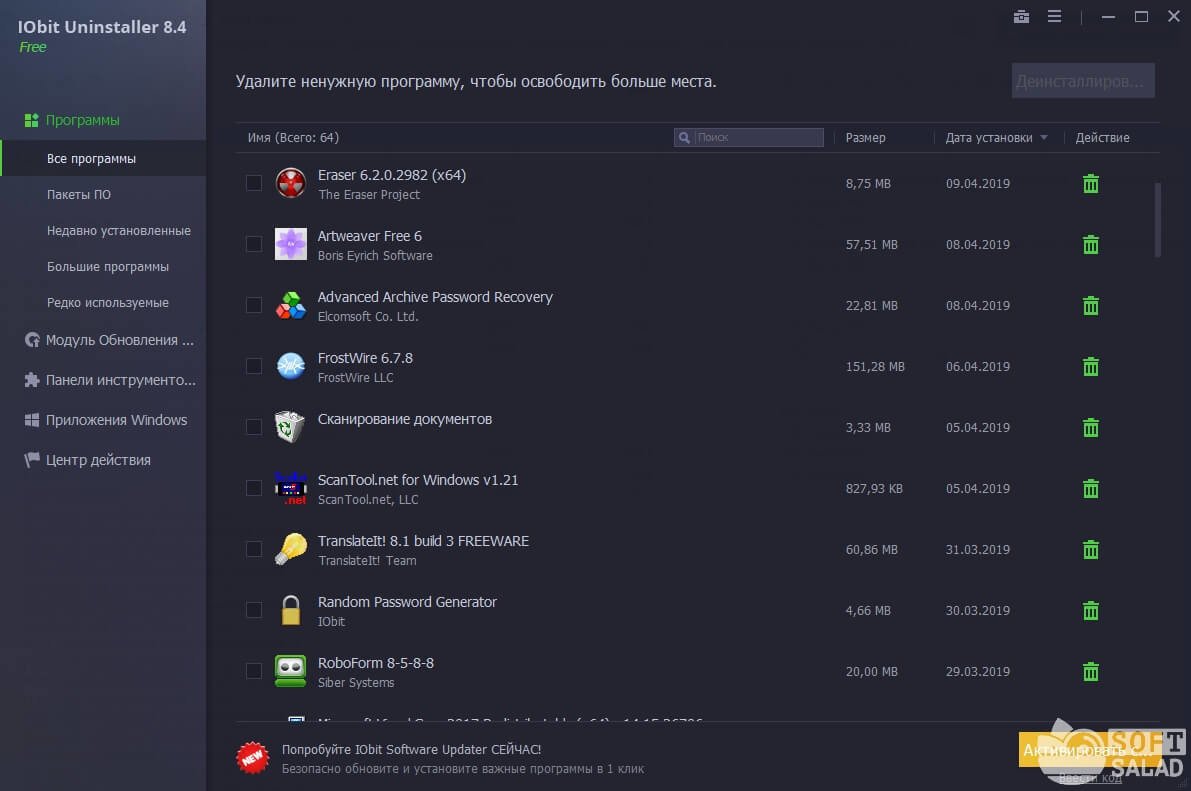


Рисунок 1.2 – Главное окно IObit Uninstaller.

Когда ваш компьютер заполняется, BleachBit быстро освобождает дисковое пространство. Когда ваша информация – это только ваш бизнес, BleachBit защищает вашу конфиденциальность. С BleachBit вы можете освобождать кеш, удалять файлы cookie, очищать историю браузера, уничтожать временные файлы, удалять логи использования программ и удалять ненужные файлы, о которых вы не знали. Разработанный для систем Linux и Windows, он очищает тысячи приложений, включая Firefox, Adobe Flash, Google Chrome, Opera и другие. Помимо простого удаления файлов, BleachBit включает расширенные функции, такие как измельчение файлов для предотвращения восстановления, очистка свободного места на диске, чтобы скрыть следы файлов, удаленных другими приложениями, и очистку Firefox для ускорения работы. Более того BleachBit – утилита с открытым исходным кодом и полностью бесплатная. На рисунке 1.3 прилагается главная страница утилиты.

Достоинства:

- простой интерфейс;

- проект с открытым исходным кодом;

- полностью бесплатная утилита.

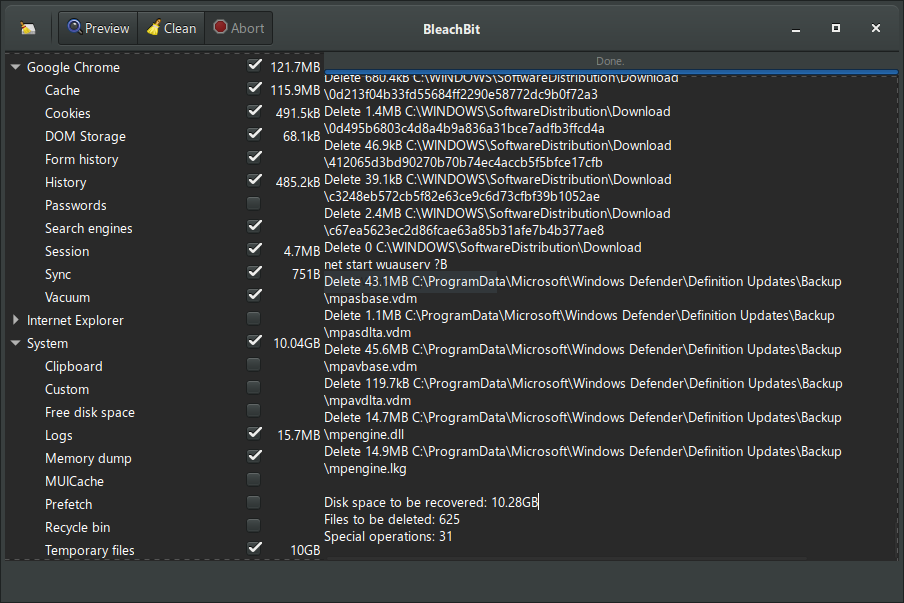


Рисунок 1.3 – Главная страница BleachBit.

1.2. Формирование требований

Целью данной курсовой работы является создание упрощённой версии утилиты CCleaner.

Используя выявленные достоинства и недостатки аналогов программы, можно определить основные требования к данному курсовому проекту, которые покрывали бы недостатки и имели достоинства рассмотренных приложений.

Из достоинств CCleaner будет использован малый вес приложения, а также понятный интерфейс. Исходя из плюсов IObit Uninstaller будет реализована поддержка старых операционных систем.

Таким образом, в ходе разработки данного программного средства планируется реализовать следующие функции:

- вычисление размера системного диска;

- поиск временных файлов;

- удаление ненужных файлов;

- поиск неиспользуемых записей в реестре;

- удаление ключей в реестре;

- очистка корзины из программы.

Для разработки программного средства будет использоваться язык программирования С++ с библиотекой WinAPI. Она предоставляет программный интерфейс к ресурсам операционной системы, таким как файловая система, реестр, графический интерфейс.

С++ был выбран ввиду следующего ряда достоинств: поддержка разных технологий и стилей программирования, включая объектно-ориентированное программирование; надежное освобождение ресурсов путем автоматического вызова деструкторов при уничтожении объектов.

В качестве среды разработки будет использоваться Microsoft Visual Studio. Она анализирует код в поисках связей между символами во всех файлах и на всех языках, используемых в проекте. На основе этого анализа IDE предоставляет помощь при написании кода, удобную навигацию, проверку ошибок в коде и, конечно, рефакторинги [1].

Для управления версиями используется система контроля версий Git, которая на данный момент является самой популярной реализацией в этой области. Система контроля версий позволяет разделять разработку на ветви и вести ее продуктивнее. Также есть возможность откатывать исходный код к рабочему состоянию в случае, когда вносились изменения, нарушающие нормальную работу проекта.

2. Разработка программного средства

2.1. Структура программы

Проект данного программного средства будет включать в себя один файл project.cpp, в котором будут реализованы все необходимые функции:

- wWinMain – создание главного окна приложения;

- WindowProc – обработка сообщений поступающих к главному окну, создание остальных частей графического интерфейса, включающих кнопки, поля ввода, статический текст;

- CheckSize – вычисление общего и свободного места на системном диске Windows;

- CheckFilesCache – вычисление общего размера и поиск файлов кэша, которые хранятся на компьютере;

- DeleteFilesCache – удаление временных файлов, найденных на диске;

- EmptyBin – очистка корзины.

2.2. Пользовательский интерфейс

Для реализации пользовательского интерфейса использовались функции, предоставляемые Windows API. Точкой входа в программу является функция wWinMain, в которой создается и инициализируется оконный класс, после чего происходит запуск цикла отслеживания полученных сообщений.

При создании главного окна на дескриптор поступает команда WM\_CREATE, обработка которой проходит в методе WindowProc. В этом методе определяется команда, поступившая на обработку. В случае команды WM\_CREATE создаются все управляющие элементы: кнопки, статический вывод текста, поле вывода программы, поле ввода данных.

Реализовано это при помощи функции CreateWindow, в которой указывается тип создаваемого элемента, дескриптор родительского окна, позиция на окне и размер, внутренний текст элемента, определённые стили. Синтаксис данной функции указывается далее:

void CreateWindow(

lpClassName,

lpWindowName,

dwStyle,

x, y, nWidth, nHeight,

hWndParent,

hMenu,

hInstance,

lpParam

);

Взаимодействие с интерфейсом посылает главному окну команду WM\_COMMAND. Обработчик такого типа события находится в методе WindowProc и, в зависимости от вызвавшего его элемента, выполняется функция. Блок-схема обработки приведена на рисунке 2.1.



Рисунок 2.1 – Блок-схема WindowProc.

2.3. Определение размера системного диска

При запуске программы в самом её верху указывается общее и свободное место на системном диске пользователя. Также получить эти данные можно по нажатию кнопки Get, расположенной слева.

Реализовано это было при помощи использования функции GetDiskFreeSpaceEx[2]. Для её правильной работы необходимо передать путь к диску и указатели на переменные, в которых будет храниться результат типа ULARGE\_INTEGER. Данная функция возвращает ненулевое значение в случае успеха и 0 при ошибке.

В переменные записывается количество байт общего размера, свободного места, а также свободного места для текущего пользователя. Если выводить полученные данные в чистом виде, это будет очень громоздко и непонятно. Таким образом было принято решение выводить полученные значения в гигабайтах, для чего потребовалось приведение переменных к типу \_\_int64, деление их на константу, определяющую количество байт в одном гигабайте, а после приведение к типу double, позволяющему использовать дробные числа.

Для информирования пользователя о результате используются статические метки, созданные в WindowProc. Однако они не могут принимать численные типы, что потребовало создание строки для вывода. Это было сделано при использовании переменной типа wstringstream[3]. В неё можно поместить числовую переменную, которая будет записана в строку. Реализация данной функции представлена ниже:

std::wstringstream wssTotal(L"");

wssTotal << TotalGB;

wchar\_t outTotal[80];

wcscpy(outTotal, L"Total size of C: ");

wcscat(outTotal, wssTotal.str().c\_str());

wcscat(outTotal, L"GB");

SetWindowText(hStatSize, outTotal);

Для установки полученного значения в метку пользовательского интерфейса используется функция SetWindowText.

2.4. Поиск файлов кэша

По нажатии кнопки Check cache запускается метод CheckFilesCache, в котором происходит поиск ненужных файлов и подсчет их суммарного размера.

На системном диске расположено множество файлов, большинство из которых необходимо для правильной работы операционной системы. Чтобы не затронуть эти данные, метод производит поиск в конкретно определённых папках, содержащих внутри себя мусорные данные, а именно:

- C:\Windows\Temp – папка Temp в Windows 10 предназначена для хранения временных файлов, которые необходимы для работы системы или какого-то приложения. Со временем папки Temp имеют неприятное свойство разрастаться до совсем уж неприличных размеров за счет того, что не все программы удаляют за собой свои временные файлы;

- C:\Windows\SoftwareDistribution – это папка, используемая службой обновления Windows для загрузки обновлений на компьютер с последующей их установкой, а также хранящая сведения обо всех ранее установленных обновлениях. После установки они остаются там еще некоторое время, а затем удаляются системой автоматически. Она присутствует во всех версиях Windows от XP и до 10;

- C:\inetpub\logs\LogFiles – папка Inetpub создаётся процессом IIS (Internet Information Services), входящим в состав Windows 10. Этот процесс реализует системный сервис, отвечающий за настройку интернет-серверов, передачу файлов по локальной сети и использование различных протоколов обмена данными. В папке Inetpub сервис хранит информацию об имеющихся серверах и их настройках.

Для поиска файлов в указанных директориях использовались следующие функции: FindFirstFile, FindNextFile, FindClose. В первую функцию передаётся путь к папке, в которой необходимо произвести поиск и указатель на структуру данных типа WIN32\_FIND\_DATA, в которую будет записана вся информация о найденном файле, включая размер, дату создания, изменения, последней записи, атрибуты, флаги. Возвращаемое значение является дескриптором найденного файла. Если оно равно INVALID\_HANDLE\_VALUE, требуется произвести проверку на тип последней ошибки. Выполняется это с помощью функции GetLastError, возвращающей код ошибки в переменную типа DWORD. Существует большое количество данных кодов, однако наиболее встречающийся – код 5, информирующий о недостаточных правах доступа. В таком случае программа выведет ошибку, требующую запустить утилиту с правами администратора, и продолжит свою работу по поиску оставшихся файлов. Окно ошибки показано на рисунке 2.2:

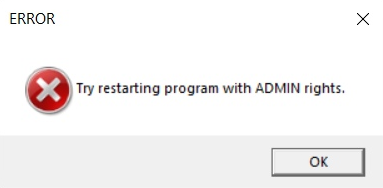


Рисунок 2.2 – Окно ошибки.

После нахождения первого файла в указанной директории, запускается цикл do … while, во время которого происходит обработка файла и переход к следующему.

Обработка заключается в следующем: сначала проверяется, является ли найденный файл директорией посредством проверки атрибута FILE\_ATTRIBUTE\_DIRECTORY в структуре данных WIN32\_FIND\_DATA. Если проверка успешна, происходит сравнение имени папки с установленными именами “.“ и “..“, которые обозначают текущую и предыдущую директории соответственно, так как поиск производится там не должен. После чего указывается новый путь поиска файлов с использованием функций wcscpy, wcscat, позволяющих определить строку типа wchar\_t, и рекурсивно вызывается метод CheckFilesCache с переданным новым путем поиска.

Если найденный файл не является директорией, из данных о нем получается время его создания и приводится из FILETIME к типу SYSTEMTIME с использованием функции FileTimeToSystemTime, в которую передаются указатели на соответствующие переменные[4]. Это необходимо для упрощения подсчета разницы во времени создания файла и текущем времени.

Далее осуществляется проверка на разницу во времени, представленная ниже:

if ((((sysTime.wYear \* 365) + (sysTime.wMonth \* 30) + sysTime.wDay) - ((fileTime.wYear \* 365) + (fileTime.wMonth \* 30) + fileTime.wDay)) > difference)

Переменная difference устанавливается пользователем после получения значения в поле ввода в интерфейсе. Если файл удовлетворяет данному условию, вычисляется его размер и записывается по адресу в переменную totalSize, содержащую общий размер найденных файлов.

Для наглядности происходящего поиска было принято решение добавления области с выводом просмотренных файлов. При вычислении размера файла также происходит добавление его полного пути в область лога. Реализовано это при помощи отправки сообщений функцией SendMessage дескриптору элемента следующим образом:

DWORD StartPos, EndPos;

SendMessage(hListCheck,EM\_GETSEL, reinterpret\_cast<WPARAM>(&StartPos), reinterpret\_cast<WPARAM>(&EndPos));

int outLength = GetWindowTextLength(hListCheck);

SendMessage(hListCheck,EM\_SETSEL,outLength, outLength);

SendMessage(hListCheck,EM\_REPLACESEL,TRUE, reinterpret\_cast<LPARAM>(rootBuff));

SendMessage(hListCheck, EM\_SETSEL, StartPos, EndPos);

После просмотра всех файлов в директории вызывается функция FindClose, завершающая работу с файлами, и проверяется код последней ошибки. При нормальной работе программы последним будет ERROR\_NO\_MORE\_FILES, в ином случае в окно вывода устанавливается текст ERROR и завершается работа метода.

Далее в метку о размере найденных файлов устанавливается новое значение, вычисляемое из переменной totalSize, в соответствии с найденным размером: гигабайты, мегабайты, килобайты или байты.

При наличии большого количества файлов, данный метод может продолжаться длительный промежуток времени, поэтому было решено вынести его выполнение в отдельный поток исполнения. Для этого используется функция CreateThread[5]. Это позволило приложению не зависать на выполнении одной задачи, а пользовательскому интерфейсу продолжать свою работу.

Большое количество ненужных данных также хранится в директории C:\Users\\*\AppData\Local\Temp. Однако возникает проблема, заключающаяся в том, что на компьютере может быть зарегистрировано несколько пользователей. Для перебора их всех стало нужным описать дополнительную функцию CheckFilesCacheUsers, которая почти идентична функции CheckFilesCache, за исключением следующих моментов: файлы, которые не являются директорией, пропускаются, при нахождении папки в ней вызывается функция CheckFilesCache, после поиска всех директорий в указанной не изменяется окно вывода на пользовательском интерфейсе. Блок-схема метода CheckFilesCacheUsers описана на рисунке 2.3.



Рисунок 2.3 – Блок-схема CheckFilesCacheUsers.

2.5. Удаление ненужных файлов

По нажатии кнопки Delete found files запускается метод ThreadDeleteCache, в котором происходит поиск ненужных файлов, подсчет их суммарного размера и удаление.

Данная функция также выполняется в отдельном потоке, чтобы не тормозить выполнение основного окна, что негативно влияет на мнение пользователя об использовании.

Перед началом выполнения процедуры очистки появляется окно с подтверждением. В нем указывается, что данное действие невозможно отменить и что пользователю следует проверить список найденных файлов. Внешний вид окна указан на рисунке 2.4.

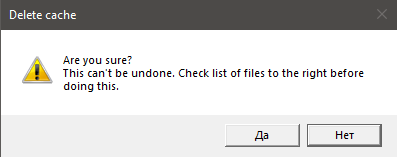


Рисунок 2.4 – Внешний вид окна предупреждения.

При выборе ответа Нет происходит возврат на главное окно программы и удаление не запускается. В обратном случае программа очищает вывод лога справа и начинает поиск файлов по директориям, содержащим в себе мусорные файлы. Для этого по порядку вызываются методы DeleteFilesCache для каждой из папок. Поиск файлов идентичен описанному в пункте 2.4.

Во время поиска, при обнаружении файла, который не является директорией, сначала происходит попытка его удаления, а только после этого высчитывается размер и полный путь записывается в логи. Удаление содержит в себе несколько шагов:

- создание переменной, содержащей в себе путь к удаляемому файлу;

- изменение атрибутов файла на стандартные;

- удаление файла с диска;

На каждом из шагов проверяется его успешность. Для этого используется функция GetLastError. Если она возвращает код 5, обозначающий недостаточные права доступа, пользователю выводится окно, требующее перезапуск программы с правами администратора.

В файловой системе также находятся файлы, которые используются другим приложением в данный момент. В таком случае кодом ошибки становится 32 - ERROR\_SHARING\_VIOLATION. Такие файлы программа не может удалить, поэтому они пропускаются, не засчитываются в общий размер и не выводятся в логи.

Программа может не найти файл по указанному пути, в таком случае кодом ошибки станет 2 - ERROR\_FILE\_NOT\_FOUND. Это указывает на установку неправильного пути к файлу.

Изменение атрибутов файла необходимо, так как есть свойство Read-Only, которое запрещает изменение файла в любом его виде. Тем самым делает невозможным очистку диска от него.

После того, как программа очистила все возможные файлы в директории, происходит проверка на её пустоту. Если папка не пуста и в ней остались файлы, удаление не происходит и программа возвращает результат. В ином случае осуществляется попытка установления необходимых атрибутов при помощи SetFileAttributes и FILE\_ATTRIBUTE\_NORMAL, удаления директории с использованием функции RemoveDirectory. Также на каждом шаге проверяется статус выполнения. Функции возвращают ненулевое значение при успехе, а при неудаче – 0. В таком случае определяется код ошибки и выполняется её обработка.

Далее программа помечает папку как удалённую и заносит в окно вывода справа на главном окне, обновляет общий размер удалённых файлов и изменяет метку справа от кнопки Delete found files в соответствии с размером: гигабайты, мегабайты, килобайты, байты. Алгоритм определения размера заключается в следующем: полученное число байт делится на количество содержащихся в одном мегабайте байт, если результат превышает это количество, то ответ будет записан в гигабайтах, и так далее.

При написании методов удаления также возникла проблема с указанной директорией: C:\Users\\*\AppData\Local\Temp. При этом возникает проблема, заключающаяся в том, что на компьютере может быть зарегистрировано несколько пользователей, а очистку необходимо производить для всех сразу. В результате была написана функция DeleteFilesCacheUsers, которая аналогична функции CheckFilesCacheUsers, за исключением того, что при нахождении директории вызывается DeleteFilesCache.

2.6. Очистка корзины

Пользователь может очистить корзину своей операционной системы путем нажатия кнопки Empty recycle bin. Корзина в операционной системе Windows создана для того, чтобы резервировать удаляемые данные на тот случай, если вдруг удаление было случайным или поспешным. Осознание этого может прийти не сразу, а через день, два или даже неделю. В этом случае всегда можно обратиться к Корзине и восстановить удаленный документ. Иными словами, это своего рода система резервного копирования, которая на время откладывает удаление документа или папки с диска, давая пользователю возможность вернуть данные в случае необходимости.

Иногда корзина может переполняться, что приводит к замедлению работоспособности всего компьютера, поэтому нужно регулярно её очищать от скопившихся в ней данных.

По нажатию кнопки у пользователя запросится подтверждение его действий и будет рекомендовано проверить корзину на наличие важных файлов. Окно подтверждения перед очисткой корзины прилагается на рисунке 2.5.

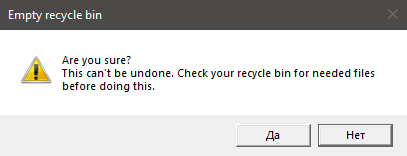


Рисунок 2.5 – Окно подтверждения перед очисткой корзины

При выборе варианта ответа Да начнётся очистка корзины как при очистке её вручную. Достигается это при помощи использования встроенной функции SHEmptyRecycleBin. Проверяется возвращаемое значение, если оно равно S\_OK, в метке справа от кнопки устанавливается положительный результат, иначе ERROR.

2.7. Взаимодействие пользователя с интерфейсом

Для улучшения опыта использования программы было принято решение отключать некоторые элементы во время её работы, а именно:

- при запуске программы отключена кнопка Delete found files для предотвращения удаления файлов, с которыми пользователь не ознакомлен;

- по нажатию кнопки Check cache она отключается, что не даёт пользователю нарушить работу программы и получить недостоверные данные. Такое состояние программы указано на рисунке 2.6;

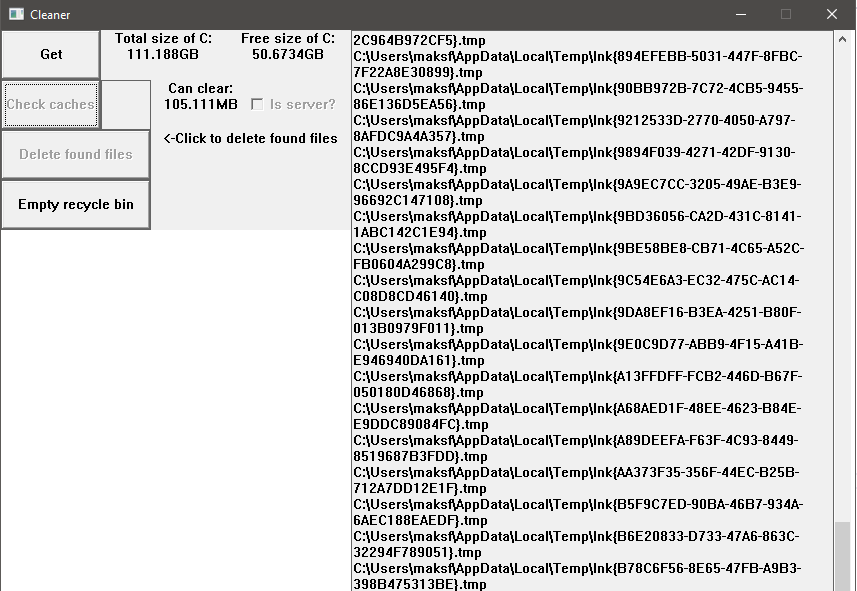


Рисунок 2.6 – Состояние программы после нажатия кнопки Check cache.

- по окончании работы функции ThreadCheckCache открываются кнопки Check caches и Delete found files, что позволяет пользователю удалить найденные файлы. Состояние прилагается далее на рисунке 2.7.

Всё это реализовано при помощи функции EnableWindow, в которую передаётся дескриптор окна и булевская переменная, определяющая состояние.

При запуске программы в пустые метки устанавливаются подсказки по использованию, которые позже будут заменены на полученные данные. Это позволяет уменьшить нагруженность пользовательского интерфейса и сделать его более минималистичным. Текст меток представлен на рисунке 2.8.

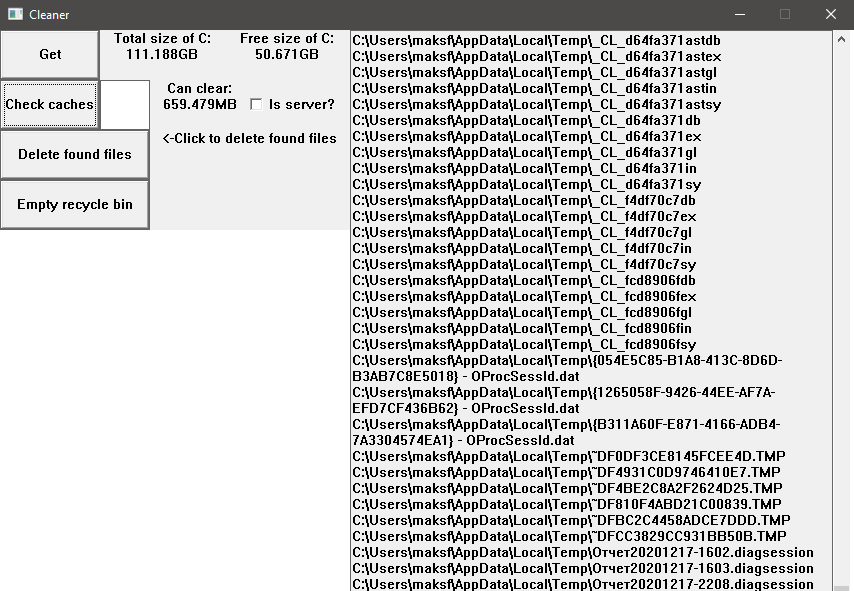


Рисунок 2.7 – Состояние программы по окончанию работы функции ThreadCheckCache

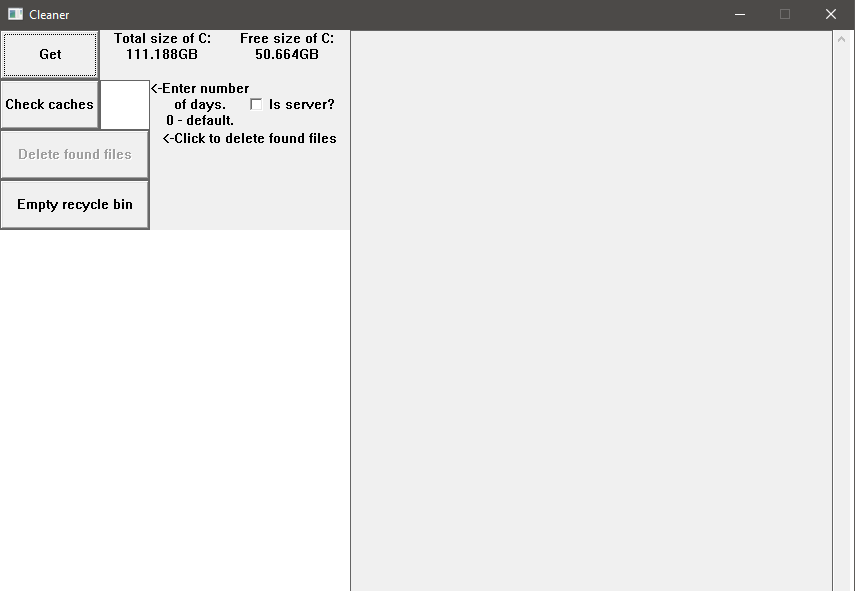


Рисунок 2.7 – Текст меток

3. Руководство пользователя

3.1. Порядок использования

При запуске программы автоматически выводится размер системного диска и количество свободного пространства на нём же. Для начала работы требуется ввести количество дней, которые будут выражать разницу между временем создания файла, который можно удалить, и системным временем, выбрать является ли компьютер сервер установкой галочки справа от поля ввода и нажать кнопку Check cache, которая начнёт определение файлов, возможных для очистки. Справа от неё будет выведен общий размер найденных файлов и полный их список. Пример на рисунке 3.1.

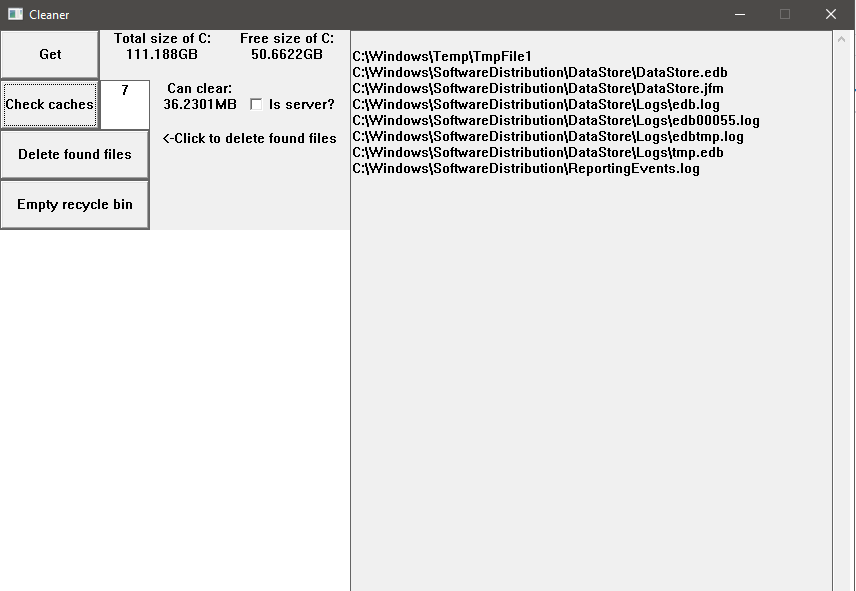


Рисунок 3.1 – Пример работы Check cache.

После этого пользователь может нажать кнопку Delete found files. Появится окно подтверждения, предупреждающее пользователя об полной утере данных для удаления. По нажатию кнопки Да в выбранном меню начнется очистка системного диска от мусорных файлов.

При нажатии кнопок Check caches или Delete found files может появиться предупреждение о некорректной работе программы, ввиду отсутствия прав администратора. В таком случае пользователь должен перезапустить программу нажатием правой кнопкой мыши на исполняемый файл и выбором Запуск от имени администратора. Окно с предупреждением прилагается ниже на рисунке 3.2.

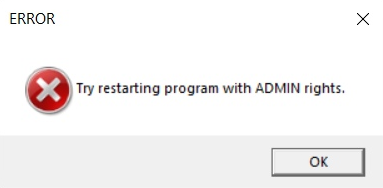


Рисунок 3.2 – Окно с предупреждением.

После удаления найденных файлов окно программы выглядит следующим образом:



Рисунок 3.3 – Окно программы после удаления.

Также, по желанию, пользователь может очистить корзину нажатием кнопки Empty recycle bin, расположенной под остальными элементами. Возникнет окно подтверждения, после которого произойдет очистка корзины операционной системы. Внешний вид прилагается на рисунке 3.4.

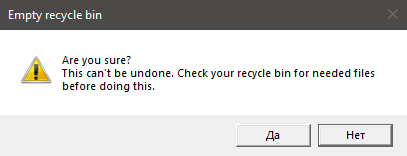


Рисунок 3.2 – Внешний вид окна.

Заключение

В рамках данного курсового проекта было разработано программное средство Cleaner. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

- вычисление размера системного диска;

- поиск временных файлов;

- удаление ненужных файлов;

- очистка корзины из программы.

Для успешного создания данного программного средства было необходимо изучить возможности среды программирования Microsoft Visual Studio, изучить теорию работы с файловой системой, особенности языка программирования С++.

Существует множество способов дальнейшего улучшения приложения. Например, добавление очистки реестра, планирование очистки пользователем, улучшение точности алгоритмов поиска файлов, добавление очистки несистемных разделов.

Данное приложение позволит держать жесткий диск и реестр системы в чистоте, что приводит к улучшению производительности персонального компьютера пользователя.

Список использованных источников

[1] Руководство по Microsoft Visual Studio. – Режим доступа https://visualstudio.microsoft.com/ru/vs/getting-started/

[2] Документация по использованию WinApi. – Режим доступа https://docs.microsoft.com/en-us/windows/win32

[3] Руководство по языку программирования С++. – Режим доступа https://metanit.com/cpp/tutorial/

[4] Обзор файловых систем FAT, HPFS и NTFS. – Режим доступа https://support.microsoft.com/ru-ru/help/100108/overview-of-fat-hpfs-and-ntfs-file-systems

[5] Win32 API Tutorial. – Режим доступа http://www.winprog.org/tutorial/

Приложение А

**Исходный код метода CheckFilesCache**

void CheckFilesCache(HWND hStatCheck, LARGE\_INTEGER \*totalSize, wchar\_t\* root, int difference) {

WIN32\_FIND\_DATA ffd;

HANDLE hFind = INVALID\_HANDLE\_VALUE;

LARGE\_INTEGER fileSize;

DWORD dwError = 0;

wchar\_t buff[250];

wcscpy(buff, root);

wcscat(buff, L"\\\*");

hFind = FindFirstFile(buff, &ffd);

if (hFind == INVALID\_HANDLE\_VALUE)

{

dwError = GetLastError();

if (dwError == 5) {

SetWindowText(hStatCheck, L"ERROR");

MessageBox(NULL, L"Try restarting program with ADMIN rights.", L"ERROR", MB\_OK | MB\_ICONERROR);

}

return;

}

do

{

if (ffd.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY)

{

if ((wcscmp(ffd.cFileName, L".")) && (wcscmp(ffd.cFileName, L".."))) {

wchar\_t rootBuff[250];

wcscpy(rootBuff, root);

wcscat(rootBuff, L"\\");

wcscat(rootBuff, ffd.cFileName);

CheckFilesCache(hStatCheck, totalSize, rootBuff, difference);

}

}

else

{

SYSTEMTIME sysTime, fileTime;

GetSystemTime(&sysTime);

FileTimeToSystemTime(&ffd.ftCreationTime, &fileTime);

if ((((sysTime.wYear \* 365) + (sysTime.wMonth \* 30) + sysTime.wDay) - ((fileTime.wYear \* 365) + (fileTime.wMonth \* 30) + fileTime.wDay)) > difference) {

fileSize.LowPart = ffd.nFileSizeLow;

fileSize.HighPart = ffd.nFileSizeHigh;

(\*totalSize).QuadPart += fileSize.QuadPart;

wchar\_t rootBuff[500];

wcscpy(rootBuff, L"\r\n");

wcscat(rootBuff, root);

wcscat(rootBuff, L"\\");

wcscat(rootBuff, ffd.cFileName);

DWORD StartPos, EndPos;

SendMessage(hListCheck, EM\_GETSEL, reinterpret\_cast<WPARAM>(&StartPos), reinterpret\_cast<WPARAM>(&EndPos));

int outLength = GetWindowTextLength(hListCheck);

SendMessage(hListCheck, EM\_SETSEL, outLength, outLength);

SendMessage(hListCheck, EM\_REPLACESEL, TRUE, reinterpret\_cast<LPARAM>(rootBuff));

SendMessage(hListCheck, EM\_SETSEL, StartPos, EndPos);

}

}

} while (FindNextFile(hFind, &ffd) != 0);

dwError = GetLastError();

if (dwError != ERROR\_NO\_MORE\_FILES)

{

SetWindowText(hStatCheck, L"ERROR");

return;

}

FindClose(hFind);

double totalSizeData;

wchar\_t out[80];

wcscpy(out, L"Can clear: ");

std::wstringstream wss(L"");

if (((\*totalSize).QuadPart / 1048576) > 1024) {

totalSizeData = (\*totalSize).QuadPart / 1073741824.0;

wss << totalSizeData;

wcscat(out, wss.str().c\_str());

wcscat(out, L"GB");

} else if (((\*totalSize).QuadPart / 1024) > 1024) {

totalSizeData = (\*totalSize).QuadPart / 1048576.0;

wss << totalSizeData;

wcscat(out, wss.str().c\_str());

wcscat(out, L"MB");

} else if ((\*totalSize).QuadPart > 1024) {

totalSizeData = (\*totalSize).QuadPart / 1024;

wss << totalSizeData;

wcscat(out, wss.str().c\_str());

wcscat(out, L"KB");

} else {

totalSizeData = (\*totalSize).QuadPart;

wss << totalSizeData;

wcscat(out, wss.str().c\_str());

wcscat(out, L"B");

}

SetWindowText(hStatCheck, out);

}