МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное Государственное Бюджетное Образовательное Учреждение Высшего Образования “Армавирский Государственный Педагогический Университет”

Кафедра Прикладной информатики, математики и физики

**КУРСОВАЯ РАБОТА**

**по дисциплине: «Операционные системы»**

**на тему:**

**«Программа построения графической карты разделов диск»**

Выполнил:

студент 4 курса ВМ-ИВТ-4-1

Оганесян А.М.

Преподаватель:

старший преподаватель

Лапшин Н.А.

Армавир, 2020

**Оглавление:**

Глава 1. Введение и постановка задачи.

Глава 2. Анализ существующих систем.

Глава 3. Комплекс используемых программ и обоснование выбора среды проектирования.

Глава 4. Разработка программного приложения.

Глава 5. Выводы.

Глава 6. Список использованных источников.

Глава 7. Приложения.

**Глава 1. Введение и постановка задачи:**

Данная работа предлагает одно из решений проблемы создания собственной программы построения графической карты диска.

Объектом исследования является разработка системы сканирования папок и файлов с собственным интерфейсом и графиком для визуализации данных.

Предмет исследования – программа анализа и построения карты диска.

Целью курсовой работы является разработка программного модуля, обеспечивающего анализ путей к файлам и их данных с возможностью графического отображения.

Для раскрытия поставленной цели решается ряд конкретных задач, а именно:

Описание технологии создания и функционирования программного модуля;

* Разработка функциональной модели системы;
* Разработка структуры программы;
* Разработка программного приложения.

Решение поставленных задач требует привлечения следующих методов: анализа соответствующей литературы по информатике, программированию, учебников и учебных пособий, обобщения и систематизации способов реализации. Возможности использования электронных справочников определяют практическую значимость курсовой работы.

При создании сервиса должны быть учтены следующие основные требования:

* оптимальное для пользователя представление исходной и конечной информации;
* простота и удобство в эксплуатации сервиса.

Поэтому, в качестве результатов курсовой работы выступают:

1. описание структуры программы анализа дисков;
2. разработка программных модулей;
3. возможность просмотра результатов анализа.

**Глава 2. Анализ существующих систем:**

Почему не стоит использовать уже существующие системы?

1. Уникальность.

Когда вы скачиваете шаблон или покупаете его из общего доступа он, конечно же не удаляется. Поэтому вы вполне можете наткнуться на точно такую же систему в интернете. В контексте развития бизнеса и бренда это не несет вам преимуществ.

1. Брендинг.

В продолжение темы готовые решения не позволяют вам формировать собственный бренд в глазах аудитории. Может быть вам покажется это не таким важным фактором, но в долгосрочной перспективе это сильно влияет на успех.

1. Бизнес-ориентированность.

Кастомные решения хороши тем, что проводится бизнес-анализ для компании, выявляются потребности и проект затачивается конкретно для вас и ваших клиентов. Вам и вашим посетителям будет удобно и комфортно работать. Даже адаптивный шаблон (корректно отображается на мобильных устройствах) не позволит вам настроить все под себя, функциональность ограничена.

1. Интеграция.

Когда мы говорим о бизнесе, то кроме того вы работаете еще со многими инструментами: CRM, ERP-системы, социальные сети и другие вспомогательные сервисы. Очень полезно интегрировать их с сайтом, чего бесплатные шаблоны позволить себе не могут.

1. Юзабилити или же удобность использования.

Продолжая тему удобства для пользователей, разрабатывая по не по шаблону, специалисты могут правильно расположить CTA-элементы, различные блоки, которые будут привлекать клиентов и давать им нужную информацию. К тому же шаблоны не позволяют добавлять уникальные элементы, верстать текст, изображения, инфографики в современных стилях, которые будут интересны пользователям, как например разные виды слайдеров, анимации, динамики.

Всегда важно думать об удобстве клиента и о том, за сколько шагов он сможет достичь своих целей.

1. Конкуренция.

Если вы занимаетесь электронной коммерцией, то у вас может быть много конкурентов, которые предлагают аналогичные товары или услуги. Качество по играет важную роль, кастомизированность поможет вам удерживать посетителей и формировать их лояльность.

1. Тенденции.

Шаблоны довольно быстро устаревают и адаптировать их под новые тренды в дизайне или функциональности практически невозможно. Кастомизированное решение легко отредизайнить, добавить небольшую фичу, чтобы повысить интерес к ПО.

1. Юридические вопросы.

Покупая шаблон, будьте бдительны и проверьте следующую информацию, чтобы избежать проблем:

* кто обладает правами на шаблон
* можно ли вносить изменения
* нужно ли добавлять ссылки на авторов шаблона
* можно ли переводить шаблон на другой язык (важно, если вы хотите мультиязычное ПО).

Именно поэтому создание собственного ПО более перспективно нежели использование готовых шаблонов.

**Глава 3. Комплекс используемых программ и обоснование выбора среды проектирования:**

Для создания данного проекта использовался язык Python. Но почему именно он? За годы своего существования Python развился в динамичный, гибкий и очень мощный язык программирования. Многие разработчики предпочитают использовать для работы именно его, а не такие традиционные варианты, как C++ и Java. И среди разработчиков Python тоже очень популярен. Но поскольку сообщество не имеет единого мнения о том, насколько эффективно использовать Python для создания богатого функционалом ПО, давайте взвесим плюсы и минусы такого выбора.

Только сперва давайте обратим внимание на ведущие сайты и популярные веб-платформы, успешно использующие Python:

* **Instagram**. Эту невероятно популярную социальную сеть ежедневно посещают больше 4 млн. человек. Основная технология, которая использовалась при ее создании — Python.
* **Spotify**. Это приложение задает тон в индустрии стриминга музыки. Создано с использованием Python.
* **Disqus**. При помощи этого плагина пользователи ежемесячно оставляют 50 миллионов комментариев на сайтах. Основной язык программирования, использовавшийся при создании этого плагина, — Python.
* **YouTube**. Крупнейший видео хостинг YouTube является частью компании Google, которая во многих своих проектах использует язык Python и фреймворк Django.

Если посмотреть на приведенные примеры, становится ясно, что Python прекрасно подходит для создания самых разнообразных проектов. И совсем не удивительно, что он используется для различных направлений, включая путешествия, администрирование, образование, развлечения, здравоохранение, социальные медиа.

Но несмотря на многочисленные достоинства Python, многие разработчики все-таки для своих проектов выбирают другие языки программирования. Поэтому мы будем рассматривать не только плюсы, но и минусы использования Python.

Преимущества использования Python для разработки:

* **Простота использования**. Одна из самых привлекательных черт языка Python — невероятная простота использования в проектах. Дальше мы расскажем, с чем это связано.
* **Легкость изучения**. Этот язык программирования изучить не сложнее, чем самый обиходный английский. Простой синтаксис языка обеспечивает очень пологую кривую обучаемости.
* **Отлично подходит для визуализации данных**. Когда речь заходит о представлении данных на сайте или в приложении, Python просто потрясающе эффективен, и разработчики не могли этого не оценить. С его помощью можно легко создавать простые и понятные отчеты и визуальные представления данных.
* **Легкость чтения**. Простой синтаксис Python обеспечивает разработчикам отличную читаемость кода. Благодаря этому разработчикам, занятым в проекте, проще коммуницировать и обмениваться знаниями.
* **Модульность**. Для Python существует очень много различных модулей которые заметно упрощают работу в нём.
* **Несравненная гибкость**. Python это очень гибкий язык программирования. Он не замыкается в собственном мире и в процессе программирования способен смешиваться с разными языками, образуя новые реализации. Например, CPython — версия Python с C, IronPython создан для совместимости с .NET и C#, а проект PyObjc позволяет совместно использовать инструменты как Python, так и ObjectiveC. И это лишь несколько примеров.
* **Асинхронное программирование**. Все мы знаем, что шаблоны асинхронного программирования помогают решать многие проблемы, с которыми время от времени сталкиваются разработчики. Python поддерживает асинхронный код. А когда разные части программы могут работать отдельно друг от друга, это помогает выявлять и решать возникающие проблемы гораздо быстрее.

Минусы использования Python в разработке:

* **Ограничения скорости**. Поскольку Python это интерпретируемый язык программирования, программы на нем работают медленнее, чем написанные на компилируемых языках.
* **Проблемы с потоками**. Global Interpreter Lock (GIL) допускает выполнение только одного потока в каждый отдельный момент. Это создает заметные ограничения для использования языка Python.
* **Простота**. Хотя простота, на первый взгляд, это преимущество, она также является и недостатком. У программистов, привыкших к простому синтаксису, часто возникают трудности при переходе на более сложные языки, такие как Java.

По большому счету, преимущества языка Python перевешивают его недостатки. И он, благодаря своей непревзойденной гибкости, простоте использования и модульности, для многих разработчиков остается самым предпочтительным вариантом выбора в том числе и для меня.

Для более продуктивной работы я использовал в своей работе следующий модуль:

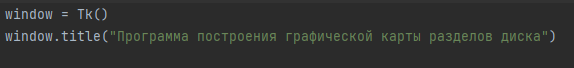
**Tkinter**. — кросс-платформенная событийно-ориентированная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX‐подобных систем, портирована также и на Microsoft Windows), написанная Стином Лумхольтом (Steen Lumholt) и Гвидо ван Россумом[1]. Входит в стандартную библиотеку Python. Tkinter плюсы:

1. Часть python, ничего лишнего для загрузки.
2. Очень простой синтаксис.
3. Текстовый виджет удивительно мощный и очень простой в работе.
4. Виджет canvas также очень прост и мощен. Ни один другой инструментарий не обеспечивает такого же сочетания простоты и мощности, как эти два виджета в моем опыте.
5. Использует собственные виджеты на mac и windows.
6. Tk - это твердая скала с несколькими кросс-платформенными особенностями. Я думаю, что механизм связывания Tkinter намного превосходит механизм wxPython; он более гибкий и последовательный. Я думаю, что три менеджера геометрии Tkinter-pack, place и grid-гораздо мощнее и проще в использовании, чем сайзеры wxPython.

**Глава 4. Разработка программного обеспечения:**

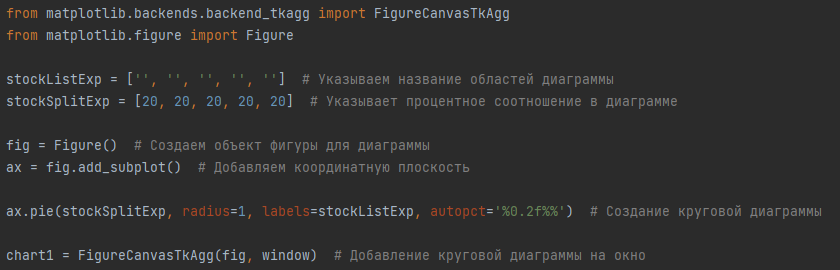
Создание проекта началось с создания графического интерфейса. Создание интерфейса можно реализовать как через код, так и через визуальную оболочку. Я же воспользовался первым вариантом.

Для начала мы инициализируем окно нашей программы (рис. 1).



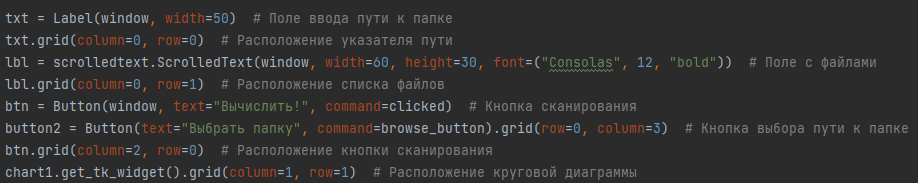
Рисунок

Затем мы создаем объект фигуры и накладываем его на координатную плоскость, затем рисуем диаграмму по заданным параметрам и добавляем ее на наше окно (рис. 2).



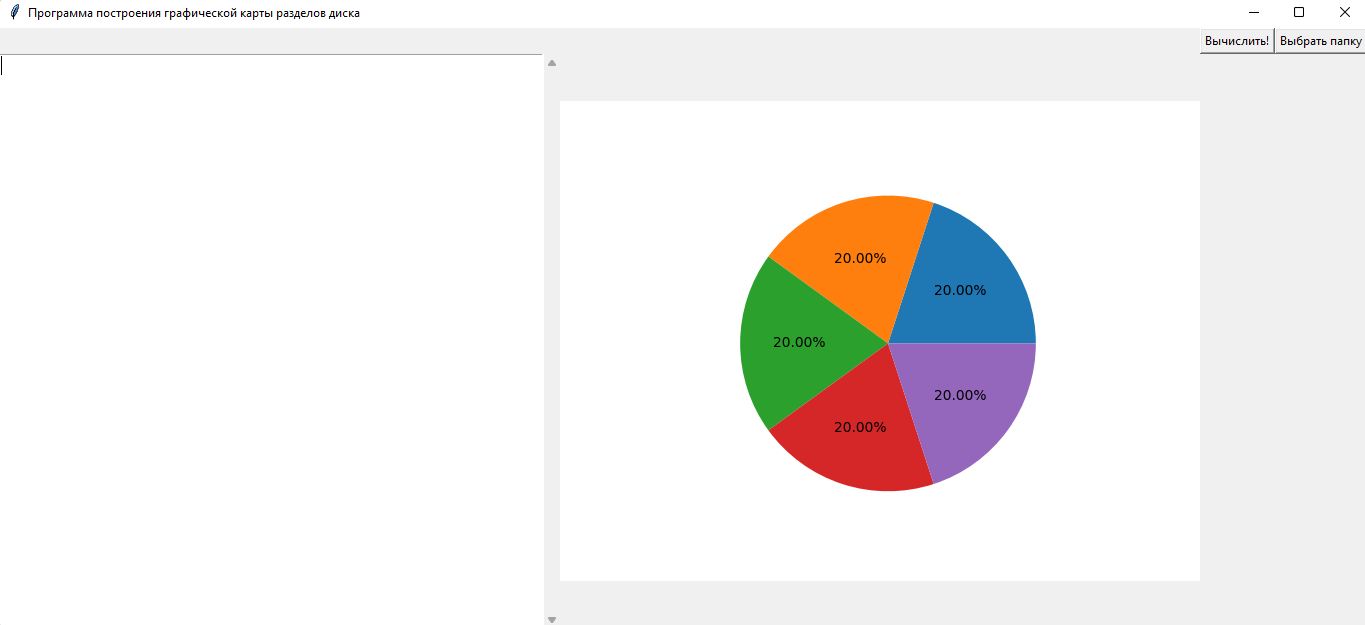
Рисунок

Следующим шагом мы инициализируем текстовые поля и кнопки в нашем окне (рис. 3).



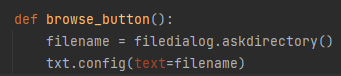
Рисунок

Так выглядит визуальный интерфейс программы, написанный через код (рис. 4).



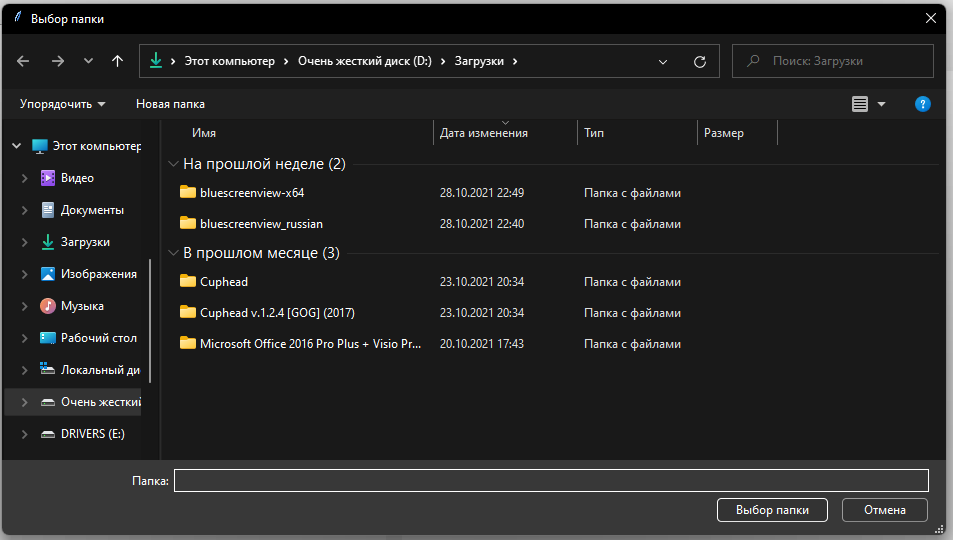
Рисунок

Добавляем кнопке выбрать папку соответствующую функцию обработки нажатия (рис. 5).



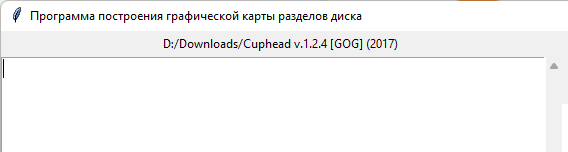
Рисунок

Так обрабатывается нажатие в визуальной части программы (рис.6).



Рисунок

После выбора папки соответствующий путь к папке появляется в левом верхнем углу программы (рис.7).



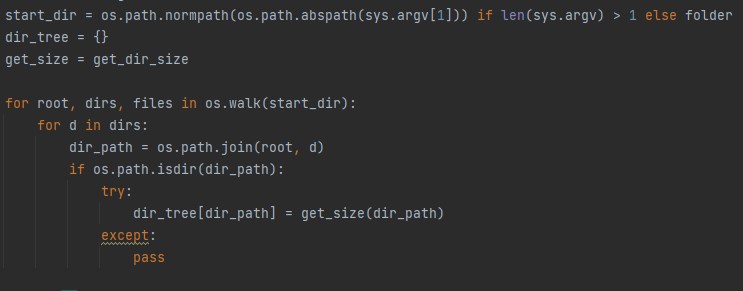
Рисунок

Далее переходим к функции обработки нажатия на кнопку “Вычислить!”, которая производит анализ папки и ее содержимого. Первым делом, программа извлекает путь из текстового окна (рис.8).



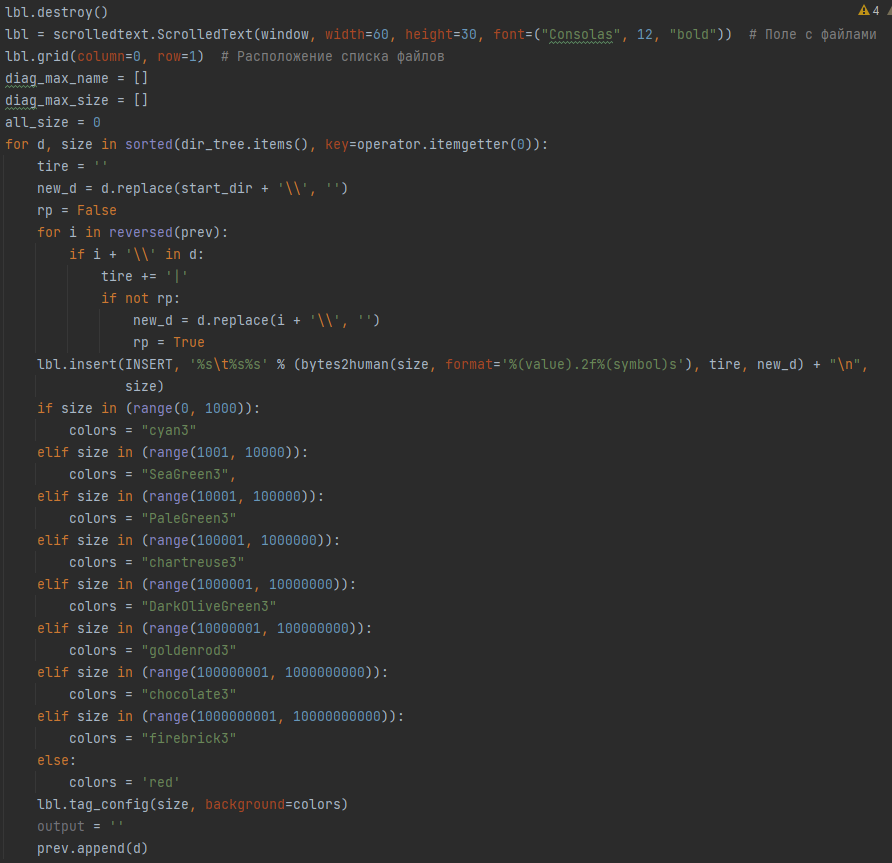
Рисунок

После определения пути мы инициализируем три массива с которыми шагаем по папкам внутри и записываем значения размера в массив, параллельно строя дерево (рис. 9).



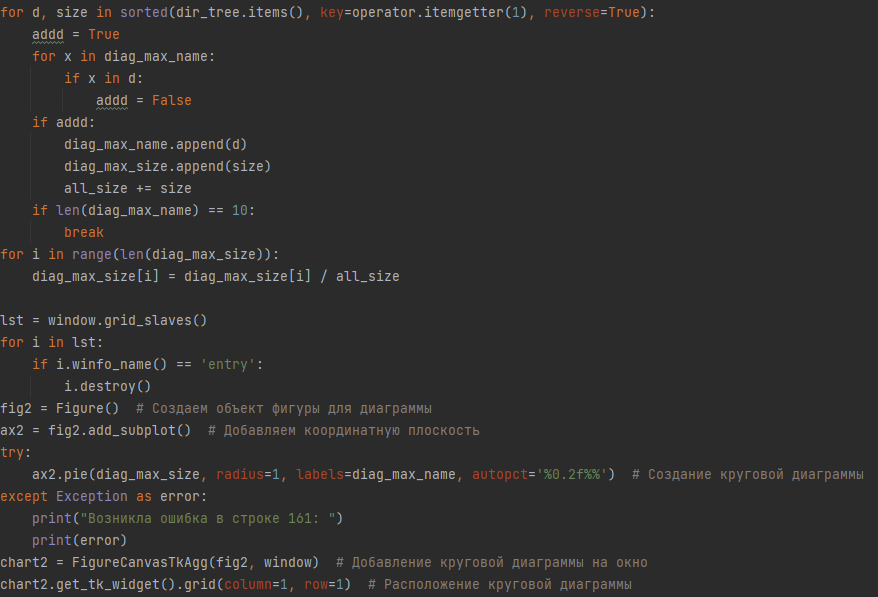
Рисунок

После анализа и построения древа, полученные данные, обрабатываются для представления в более удобной форме, а также раскрашиваются в соответствии с размером (рис. 10).



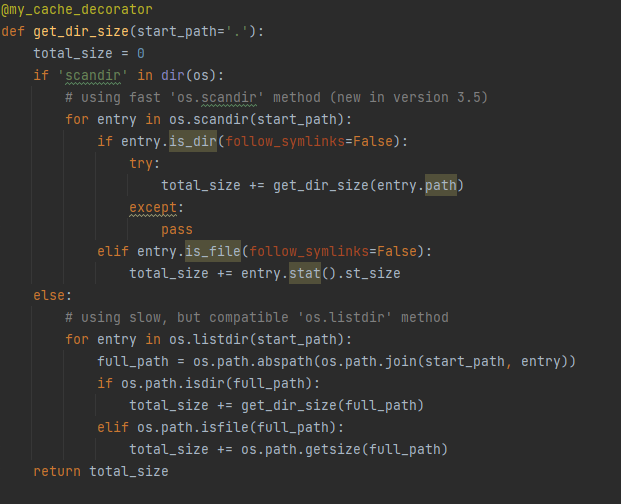
Рисунок

Затем идет сортировка размеров папок без включений и построение диаграммы (рис. 11).



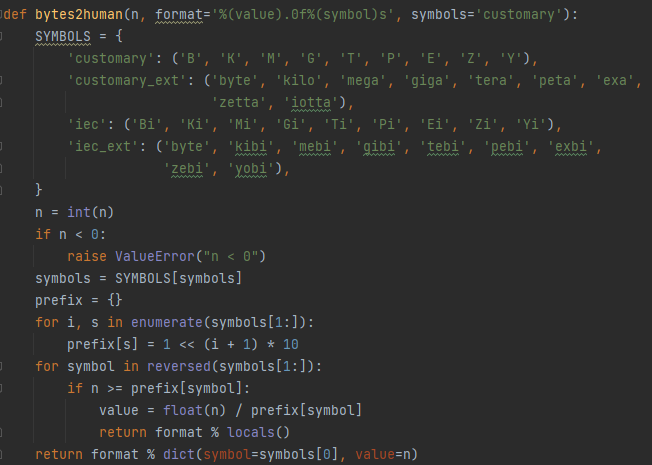
Рисунок

Сама функция определяющая размер пути выглядит так: (рис. 12).



Рисунок

Для более понятного отображения размера файлов была написана такая функция, которая в зависимости от числа добавляет соответствующую букву размера (рис.13).

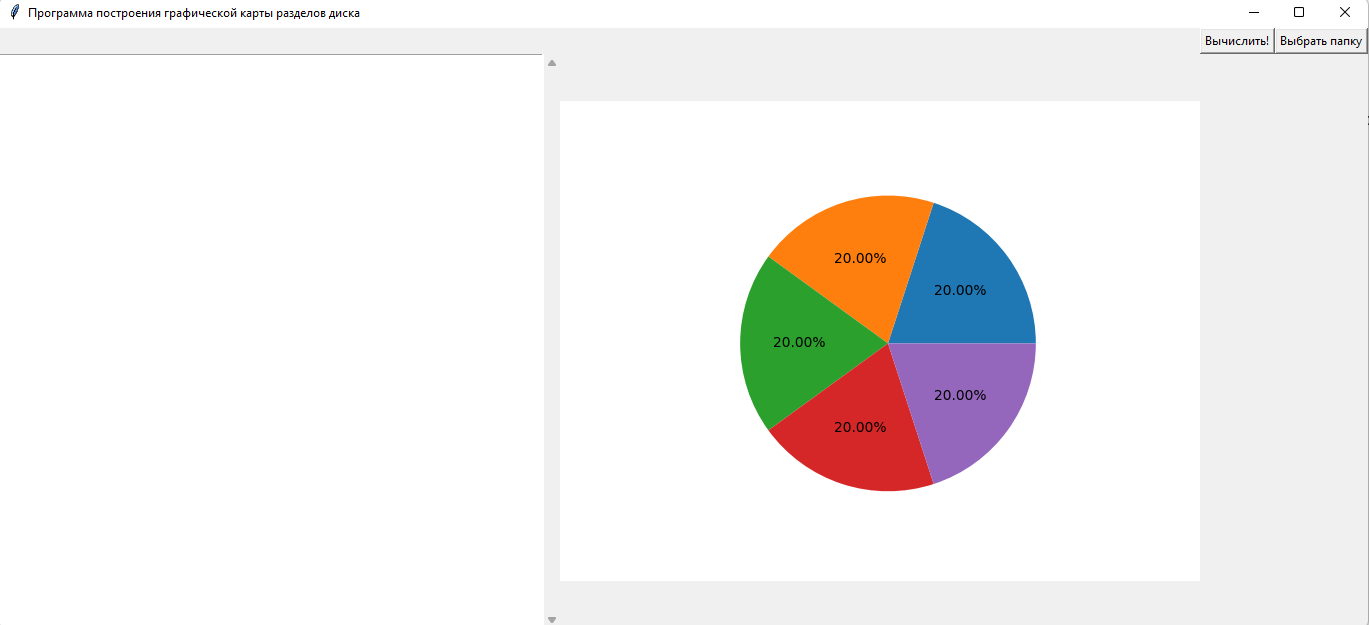


Рисунок

**Глава 5. Вывод:**

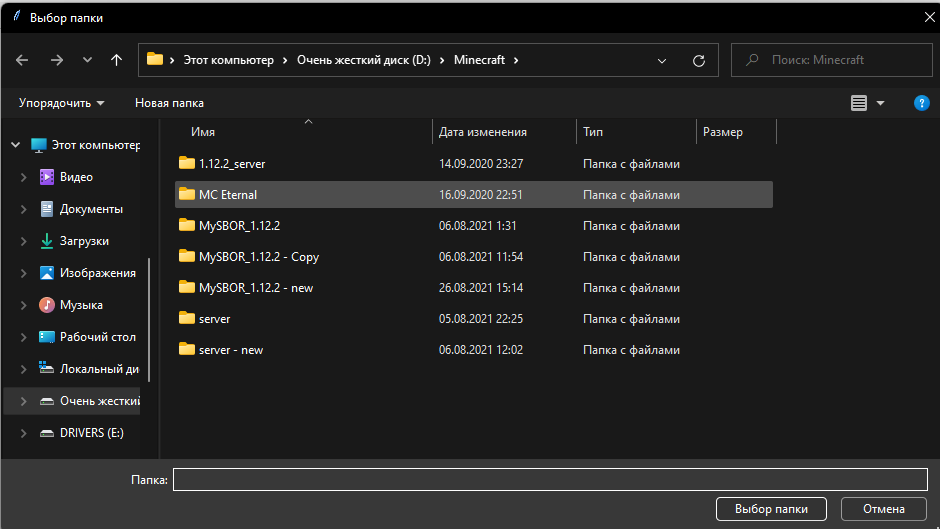
В итоге было разработано ПО с визуализацией данных, в которой вы можете просмотреть размер папок, файлов и диаграмму занимаемого места.

Финальный интерфейс программы выглядит так (рис.14):



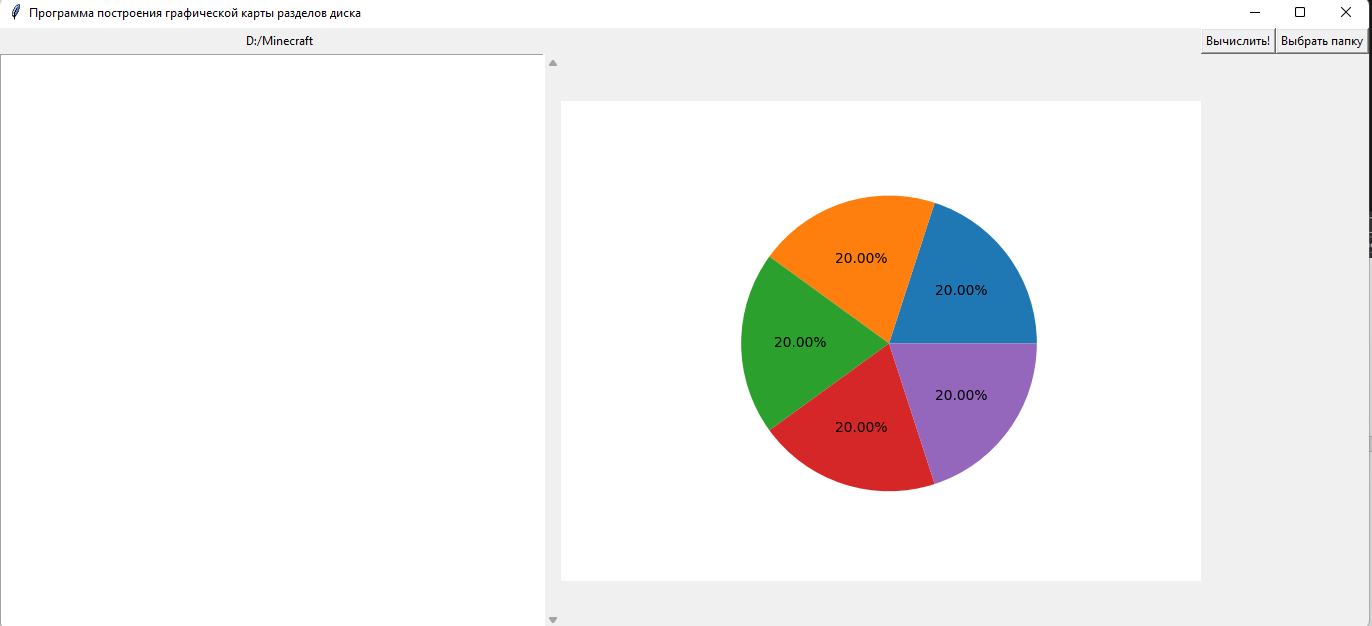
Рисунок

При выборе папки (рис. 15):



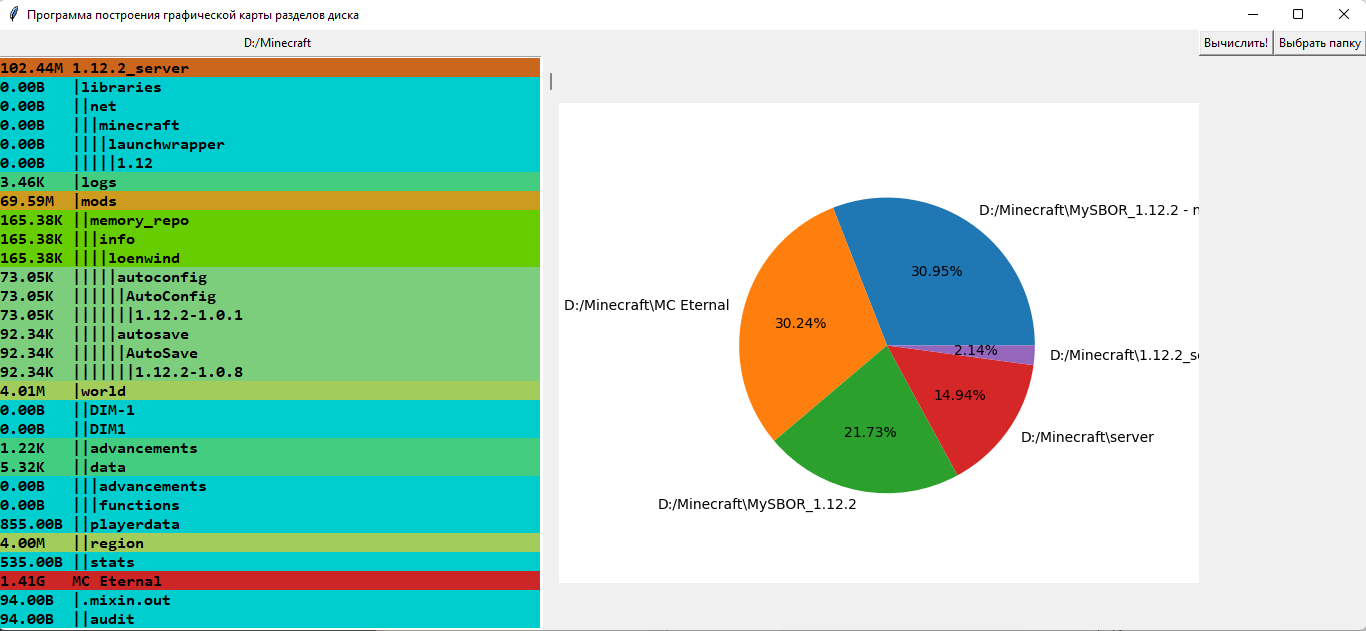
Рисунок

После выбора папки (рис. 16):



Рисунок

При анализе папки (рис.17):



Рисунок

**Список литературных источников:**

*Сайты, порталы, базы данных*

eLIBRARY.RU: научная электронная библиотека: сайт. – Москва, 2000 – . – URL: <https://elibrary.ru> (дата обращения: 01.05.2020). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

Министерство науки и высшего образования Российской Федерации: официальный сайт. – URL: [https://minobrnauki.gov.ru/](https://minobrnauki.gov.ru/%20%20)  (дата обращения: 01.05.2020). – Текст: электронный.

GitHub.com: крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. URL: <https://github.com/krutoypan3> (дата обращения: 01.05.2020). – Текст: электронный.

*Компьютерные программы*

PyCharm: интегрированная среда разработки для языка программирования Python / разработчик “JetBrains”. – Электронная программа: электронная.

**Глава 7. Приложение**

Файл main.py:

from \_\_future\_\_ import print\_function  
import os  
import sys  
import operator  
from tkinter import scrolledtext, filedialog  
  
  
def null\_decorator(ob):  
 return ob  
  
  
if sys.version\_info >= (3, 2, 0):  
 import functools  
  
 my\_cache\_decorator = functools.lru\_cache(maxsize=4096)  
else:  
 my\_cache\_decorator = null\_decorator  
  
  
@my\_cache\_decorator  
def get\_dir\_size(start\_path='.'):  
 total\_size = 0  
 if 'scandir' in dir(os):  
 # using fast 'os.scandir' method (new in version 3.5)  
 for entry in os.scandir(start\_path):  
 if entry.is\_dir(follow\_symlinks=False):  
 try:  
 total\_size += get\_dir\_size(entry.path)  
 except:  
 pass  
 elif entry.is\_file(follow\_symlinks=False):  
 total\_size += entry.stat().st\_size  
 else:  
 # using slow, but compatible 'os.listdir' method  
 for entry in os.listdir(start\_path):  
 full\_path = os.path.abspath(os.path.join(start\_path, entry))  
 if os.path.isdir(full\_path):  
 total\_size += get\_dir\_size(full\_path)  
 elif os.path.isfile(full\_path):  
 total\_size += os.path.getsize(full\_path)  
 return total\_size  
  
  
def bytes2human(n, format='%(value).0f%(symbol)s', symbols='customary'):  
 SYMBOLS = {  
 'customary': ('B', 'K', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y'),  
 'customary\_ext': ('byte', 'kilo', 'mega', 'giga', 'tera', 'peta', 'exa',  
 'zetta', 'iotta'),  
 'iec': ('Bi', 'Ki', 'Mi', 'Gi', 'Ti', 'Pi', 'Ei', 'Zi', 'Yi'),  
 'iec\_ext': ('byte', 'kibi', 'mebi', 'gibi', 'tebi', 'pebi', 'exbi',  
 'zebi', 'yobi'),  
 }  
 n = int(n)  
 if n < 0:  
 raise ValueError("n < 0")  
 symbols = SYMBOLS[symbols]  
 prefix = {}  
 for i, s in enumerate(symbols[1:]):  
 prefix[s] = 1 << (i + 1) \* 10  
 for symbol in reversed(symbols[1:]):  
 if n >= prefix[symbol]:  
 value = float(n) / prefix[symbol]  
 return format % locals()  
 return format % dict(symbol=symbols[0], value=n)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 from tkinter import \*  
  
 def clicked():  
 folder = txt.cget("text")  
 start\_dir = os.path.normpath(os.path.abspath(sys.argv[1])) if len(sys.argv) > 1 else folder  
 dir\_tree = {}  
 get\_size = get\_dir\_size  
  
 for root, dirs, files in os.walk(start\_dir):  
 for d in dirs:  
 dir\_path = os.path.join(root, d)  
 if os.path.isdir(dir\_path):  
 try:  
 dir\_tree[dir\_path] = get\_size(dir\_path)  
 except:  
 pass  
  
 prev = []  
 global lbl  
 lbl.destroy()  
 lbl = scrolledtext.ScrolledText(window, width=60, height=30, font=("Consolas", 12, "bold")) # Поле с файлами  
 lbl.grid(column=0, row=1) # Расположение списка файлов  
 diag\_max\_name = []  
 diag\_max\_size = []  
 all\_size = 0  
 for d, size in sorted(dir\_tree.items(), key=operator.itemgetter(0)):  
 tire = ''  
 new\_d = d.replace(start\_dir + '\\', '')  
 rp = False  
 for i in reversed(prev):  
 if i + '\\' in d:  
 tire += '|'  
 if not rp:  
 new\_d = d.replace(i + '\\', '')  
 rp = True  
 lbl.insert(INSERT, '%s\t%s%s' % (bytes2human(size, format='%(value).2f%(symbol)s'), tire, new\_d) + "\n",  
 size)  
 if size in (range(0, 1000)):  
 colors = "cyan3"  
 elif size in (range(1001, 10000)):  
 colors = "SeaGreen3",  
 elif size in (range(10001, 100000)):  
 colors = "PaleGreen3"  
 elif size in (range(100001, 1000000)):  
 colors = "chartreuse3"  
 elif size in (range(1000001, 10000000)):  
 colors = "DarkOliveGreen3"  
 elif size in (range(10000001, 100000000)):  
 colors = "goldenrod3"  
 elif size in (range(100000001, 1000000000)):  
 colors = "chocolate3"  
 elif size in (range(1000000001, 10000000000)):  
 colors = "firebrick3"  
 else:  
 colors = 'red'  
 lbl.tag\_config(size, background=colors)  
 output = ''  
 prev.append(d)  
  
 for d, size in sorted(dir\_tree.items(), key=operator.itemgetter(1), reverse=True):  
 addd = True  
 for x in diag\_max\_name:  
 if x in d:  
 addd = False  
 if addd:  
 diag\_max\_name.append(d)  
 diag\_max\_size.append(size)  
 all\_size += size  
 if len(diag\_max\_name) == 10:  
 break  
 for i in range(len(diag\_max\_size)):  
 diag\_max\_size[i] = diag\_max\_size[i] / all\_size  
  
 lst = window.grid\_slaves()  
 for i in lst:  
 if i.winfo\_name() == 'entry':  
 i.destroy()  
 fig2 = Figure() # Создаем объект фигуры для диаграммы  
 ax2 = fig2.add\_subplot() # Добавляем координатную плоскость  
 try:  
 ax2.pie(diag\_max\_size, radius=1, labels=diag\_max\_name, autopct='%0.2f%%') # Создание круговой диаграммы  
 except Exception as error:  
 print("Возникла ошибка в строке 161: ")  
 print(error)  
 chart2 = FigureCanvasTkAgg(fig2, window) # Добавление круговой диаграммы на окно  
 chart2.get\_tk\_widget().grid(column=1, row=1) # Расположение круговой диаграммы  
  
  
 window = Tk()  
 window.title("Программа построения графической карты разделов диска")  
  
 from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
 from matplotlib.figure import Figure  
  
 stockListExp = ['', '', '', '', ''] # Указываем название областей диаграммы  
 stockSplitExp = [20, 20, 20, 20, 20] # Указывает процентное соотношение в диаграмме  
  
 fig = Figure() # Создаем объект фигуры для диаграммы  
 ax = fig.add\_subplot() # Добавляем координатную плоскость  
  
 ax.pie(stockSplitExp, radius=1, labels=stockListExp, autopct='%0.2f%%') # Создание круговой диаграммы  
  
 chart1 = FigureCanvasTkAgg(fig, window) # Добавление круговой диаграммы на окно  
  
  
 def browse\_button():  
 filename = filedialog.askdirectory()  
 txt.config(text=filename)  
  
  
 txt = Label(window, width=50) # Поле ввода пути к папке  
 txt.grid(column=0, row=0) # Расположение указателя пути  
 lbl = scrolledtext.ScrolledText(window, width=60, height=30, font=("Consolas", 12, "bold")) # Поле с файлами  
 lbl.grid(column=0, row=1) # Расположение списка файлов  
 btn = Button(window, text="Вычислить!", command=clicked) # Кнопка сканирования  
 button2 = Button(text="Выбрать папку", command=browse\_button).grid(row=0, column=3) # Кнопка выбора пути к папке  
 btn.grid(column=2, row=0) # Расположение кнопки сканирования  
 chart1.get\_tk\_widget().grid(column=1, row=1) # Расположение круговой диаграммы  
  
 window.mainloop()