Всероссийский конкурс учебно-исследовательских работ старшеклассников
по политехническим, естественным, математическим дисциплинам
для учащихся 9-11 классов

Направление: Информатика и информационные технологии

СОЗДАНИЕ ПРОГРАММЫ «PHYSICALCALCULATION»

Выполнил:
Сырцов Михаил Юрьевич
$6^{-а}$ класс, МБОУ «Лицей № 1»,
г.Перми

Руководитель:
МБОУ «Лицей № 1», г.Перми
Брагина Наталья Алексеевна

г. Пермь, 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Персональный компьютер является уже неотъемлемой частью нашей жизни. Мы используем его для учебы, работы и просто для общения. Персональный компьютер является многофункциональным устройством, но без специальных программ многие операции было бы невозможно выполнить. Но возникает вопрос: кто и как создает программы? Вот и меня давно заинтересовал этот вопрос, а в частности - создание различных программ.

В последние 10-15 лет резко возрос интерес к программированию. Это в первую очередь связано с развитием и внедрением в повседневную жизнь информационно-коммуникационных технологий. Если человек является профессионалом в данной отрасли, то он в любом случае имел дело с программированием. Если же человек не профессионал, а возможно только интересующийся компьютерными технологиями, то рано или поздно у него возникает желание, а иногда и появляется необходимость программировать.

На современном этапе программирование может рассматриваться как наука и как искусство. Программирование представляет собой сферу действий, направленную на создание программ.

Программа - это последовательность команд компьютера, приводящая к решению задачи. Программы предназначены для машинной реализации задач. Программа является результатом интеллектуального труда, для которого характерно еще и творчество.

Множество информационных ресурсов и учебников содержат описание создания программ на различных языках программирования. Языков для программирования на ПК очень много (около 1800), все они для разных целей и отличаются друг от друга, но меня больше всего заинтересовал Python.

Python является простым и мощным объектно-ориентированным языком программирования. Python представляет структуры данных высокого уровня, синтаксис его минималистичен. В то же время стандартная библиотека включает большой объём полезных функций, что делает его идеальным языком для быстрого написания различных приложений, работающих на большинстве известных платформ. Python — активно развивающийся язык программирования, именно поэтому он был выбран мной в **моем исследовании**.

Программирование давно увлекало меня и в 8 лет я написал свою первую страницу на HTML (Рис.1). **Идея** создания программы у меня возникла с желанием научиться программировать на Python. Язык программирования Python был для меня новым, поэтому прежде чем перейти к созданию программы я изучил много литературы о данном языке.

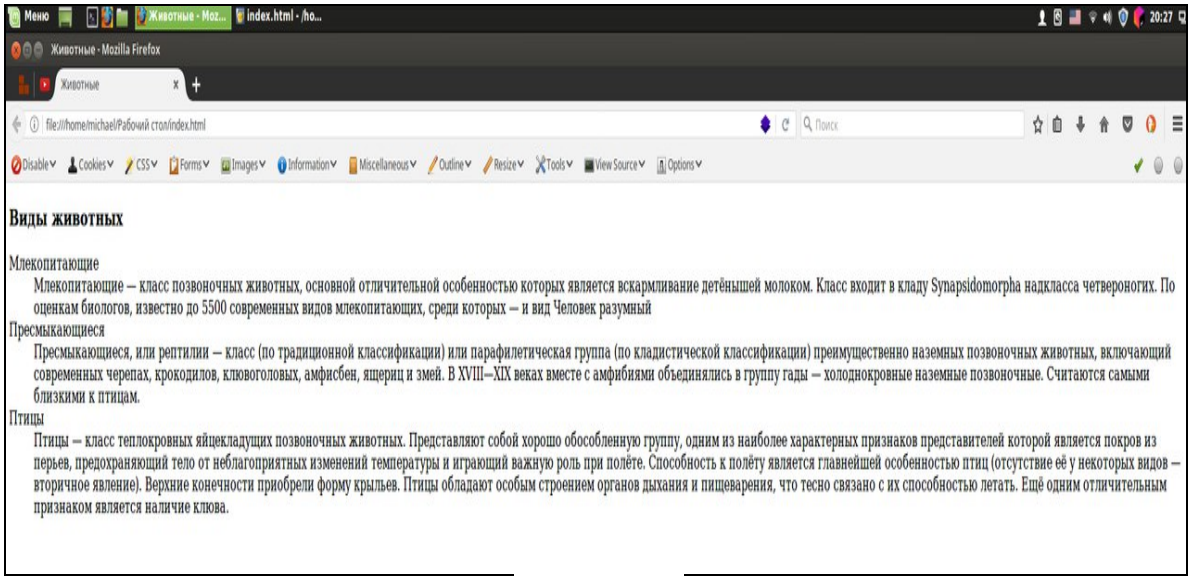


Рисунок 1

В сентябре 2017 года возникла идея создать программу, которая помогла бы выполнять различные расчеты по физике, так как из всех учебных предметов меня увлекает именно физика и информатика.

Актуальность темы в том, что разработка программы для школьного курса физики 7-8 классов является интересным как для самих учащихся так и необходимым инструментом на уроке для учителя физики.

Отсюда вытекает **гипотеза** - что создание и применение созданной программы сможет повысить интерес учащихся к физике, повысит уровень понимания решения физических задач.

Поэтому **цель** данной работы заключается в создании программы для курса школьной физики на языке программирования Python.

Для достижения цели необходимо поставить следующие **задачи**:

1. проанализировать необходимые компоненты программы для реализации расчетов по курсу физики 7-8 кл.;
2. составить схему взаимодействия модулей программы, навигацию, принцип работы;
3. определить дизайн и создать графический интерфейс для программы;
4. написать программу на языке программирования Python;
5. реализовать связующее звено между программой и графическим интерфейсом.

Методами исследования являются:

1. поиск, сбор и анализ информации по теме в различных источниках;
2. практическая реализация и применение языков программирования для создания программы.

Практическая значимость работы заключается в том, что ценность систематизированной информации позволит учащимся 7,8 классов проявить

интерес к информатике или физике, пробудить желание заниматься и развиваться не только в школе, но и дома.

Новизной в данной работе является создание программы «Physical calculation» для учащихся 7,8 классов, с помощью которой можно решать задачи по физике.

1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ СОЗДАНИЯ ПРОГРАММ С ПОМОЩЬЮ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Python представляет популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение Python получил в области машинного обучения и исследований искусственного интеллекта.

Основные особенности языка программирования Python:

1. Скриптовый язык.
2. Код программ определяется в виде скриптов.
3. Поддержка самых различных парадигм программирования, в том числе объектно-ориентированной и функциональной парадигм.
4. Интерпретация программ. Для работы со скриптами необходим интерпретатор, который запускает и выполняет скрипт.

Выполнение программы на Python выглядит следующим образом. Сначала пишем в текстовом редакторе скрипт с набором выражений на данном языке программирования. Этот скрипт выполняет интерпретатор. Интерпретатор транслирует код в промежуточный байткод, а затем виртуальная машина переводит полученный байткод в набор инструкций, которые выполняются операционной системой (Рис.2). Здесь стоит отметить, что хотя формально трансляция интерпретатором исходного кода в байткод и перевод байткода виртуальной машиной в набор машинных команд представляют два разных процесса, но фактически они объединены в самом интерпретаторе.

Рисунок 2

- Не имеет значения, какая операционная система установлена Windows, Mac OS, Linux, достаточно написать скрипт, который будет запускаться на всех этих ОС при наличии интерпретатора.
- Автоматическое управление памяти.
- Динамическая типизация

Python – очень простой язык программирования, он имеет лаконичный и в то же время довольно простой и понятный синтаксис.

Соответственно его легко изучать, и собственно это одна из причин, по которой он является одним из самых популярных языков программирования именно для обучения. Python также популярен не только в сфере обучения, но в написании конкретных программ. В немалой степени, поэтому для этого языка написано множество библиотек, которые можно использовать. Кроме того, в интернете можно найти по данному языку множество полезных материалов, примеров, получить квалифицированную помощь специалистов.

1.1. История возникновения языка программирования Python

Создание языка Python начиналось довольно медленно и неуверенно. Главным энтузиастом, который в 1990 году пытался воплотить Python в реальность, стал Гвидо Ван Россум. Именно этот человек, работая над разработкой языка ABC в Голландском институте CWI, понял, что хотел бы создать нечто новое. Это послужило стартом для написания нового интерпретатора; конечно, не без использования некоторых идей, взятых с ABC.

Интересным моментом выступает то, что первый рабочий прототип Python был создан на домашнем Макинтоше Гвидо, да и еще за пару выходных. Что касается распространения, то делалось это с помощью Интернет.

В 1996 году, когда данный проект набирал критическую массу, к разработке подключился Стив Маевский, который был довольно известным в сети, так как вел свой блог «Сравнительная критика языков программирования». Стив, как и Гвидо был поклонником Macintosh, возможно, это и послужило основой их сотрудничества. Стоить отметить, что язык получил название «Python» не в честь вида змей, как ошибочно считают многие разработчики. Во времена разработки «Питона» Гвидо любил смотреть комедийное шоу «Воздушный цирк Монти Пайтона», поэтому и назвал своей проект в честь Монти Пайтона.

Так как Питон имел отличный потенциал и свободно распространялся через Интернет, в него появилось ядро последователей - люди, которые были заинтересованы в развитии Python как язык программирования. В начале своего пути, этот язык имел вид небольшого интерпретатора с малым количеством функций и полным отсутствием ООП, что всех не устраивало и мотивировало на дальнейшее развитие языка.

Уже в 1991 году стали появляться первые средства ООП разработки.

Спустя некоторое время, Гвидо предложили должность в корпорации CNRI, которая находится в Америке. Недолго думая, Гвидо покинул Голландию и принялся за работу. Занимаясь проектами компании, он часто использовал Python для решения многих задач, а в свободное время занимался его развитием как интерпретатора.

По такой схеме Питон развивался до 1999 года и получил версию 1.5.2. По достижении этой планки в жизни Гвидо начались изменения. Компания все больше загружала его работой, что сильно уменьшало время для любимого занятия.

Это побудило задуматься Гвидо о целесообразности такой работы, в результате чего, он принял решение искать спонсора, который даст возможность работать только над развитием языка. Так как к тому времени в интернете уже существовало немалое сообщество пользователей, фирма BeOpen решила принять участие в продвижении Python. По контракту с CNRI Гвидо обязался выпустить версию 1.6, что он и сделал перед уходом. Работая с BeOpen, он показал миру версию 2.0. Многие утверждают, что версия 2.0 дала сильный толчок в социальном плане. А все потому, что процесс развития языка стал более открытым. Гвидо перевел все данные на SourceForge, что сильно понравилось сообществу, которое требовало внедрения возможности участия в разработке кода. Кроме этого в то время появился Юникод, а это большой шаг вперед. К Юникоду создали новый механизм регулярных выражений, который мог работать как с обычными строками, так и с Юникодовыми.

Через некоторое время в компании BeOpen начались проблемы. Они решили, что Гвидо должен работать усердней и приносить деньги, а не только просить их, на развитие проекта. Такое поведение не пришлось по душе Гвидо – он уволился и начал размышлять: куда идти дальше. Во многих интервью, Гвидо рассказывает, что этот фрагмент жизни был переломный.

В этот раз, свое финансирование ему предложила компания Digital Creations – авторы Zope. Как не странно, но это было пятое предложение от них, на что Гвидо согласился. В этой компании вся команда разработчиков «питона» получила большие возможности, что дало плоды. В том же году был выпущена версия 2.1. Теперь в питоне появились новые объекты с языков closures и иерархия: функции можно вкладывать друг в друга, сохраняя при этом доступ к переменным окружающих функций. Это сильно изменит язык, а главное сильно улучшит его подходы к способу программированию. На данный момент существует версия 3.5.1, что демонстрирует его развитие, ведь каждый год разработчики проделывают огромную работу. Все это превратило простой интерпретатор в очень популярный язык программирования, который используется как первый в обучении миллионов студентов по всему миру.

1.2.Плюсы и минусы языка Python

В работе была использована версия Python 3.6.3 в качестве языка для создания программы. На протяжении всей работы были выявлены плюсы и минусы данного языка. Как оказалось плюсов у данного языка очень много.

Несомненным **достоинством** является то, что интерпретатор Python реализован практически на всех платформах и операционных системах. Первым таким языком был C, однако его типы данных на разных машинах могли занимать разное количество памяти и это служило некоторым препятствием при написании действительно переносимой программы. Python же таким недостатком не обладает.

Следующая **немаловажная** черта - расширяемость языка, этому придается большое значение и, как пишет сам автор, язык был задуман именно как расширяемый. Это означает, что имеется возможность совершенствования языка всеми заинтересованными программистами. Интерпретатор написан на C и исходный код доступен для любых манипуляций. В случае необходимости, можно вставить его в свою программу и использовать как встроенную оболочку. Или же, написав на C свои дополнения к Python и скомпилировав программу, получить "расширенный" интерпретатор с новыми возможностями.

Следующее **достоинство** - наличие большого числа подключаемых к программе модулей, обеспечивающих различные дополнительные возможности. Такие модули пишутся на C и на самом Python и могут быть разработаны всеми достаточно квалифицированными программистами. В качестве примера можно привести следующие модули:

1. Numerical Python - расширенные математические возможности, такие как манипуляции с целыми векторами и матрицами;
2. Tkinter - построение приложений с использованием графического пользовательского интерфейса (GUI) на основе широко распространенного на X-Windows Tk-интерфейса;
3. OpenGL - использование обширной библиотеки графического моделирования двух- и трехмерных объектов Open Graphics Library фирмы Silicon Graphics Inc. Данный стандарт поддерживается, в том числе, в таких распространенных операционных системах как Microsoft Windows 95 OSR 2, 98 и Windows NT 4.0.

Рассмотрим так же некоторые особенности языка, которые можно отнести к **плюсам** языка.

1. Python, в отличие от многих языков (Pascal, C++, Java, и т.д.), не требует описания переменных. Они создаются в месте их инициализации, т.е. при первом присваивании переменной какого-либо значения. Значит, тип переменной определяется типом присваиваемого значения. В этом отношении Python напоминает Basic. Тип переменной не является неизменным. Любое присваивание для нее корректно и это приводит лишь к тому, что типом переменной становится тип нового присваиваемого значения.

2. В таких языках как Pascal, C, C++ организация списков представляла некоторые трудности. Для их реализации приходилось хорошо изучать принципы работы с указателями и динамической памятью. И даже имея хорошую квалификацию, программист, каждый раз заново, реализуя механизмы создания, работы и уничтожения списков, мог легко допустить трудноуловимые ошибки. Ввиду этого были созданы некоторые средства для работы со списками. Например, в Delphi Pascal имеется класс TList, реализующий списки; для C++ разработана библиотека STL (Standard Template Library), содержащая такие структуры как векторы, списки, множества, словари, стеки и очереди. Однако, такие средства имеются не во всех языках и их реализациях.

Одной из отличительных черт Python является наличие таких встроенных в сам язык структур как тьюплы (tuple), списки (list) и словари (dictionary), которые иногда называют картами (map).

Рассмотрим их поподробней:

a. Тьюпл. Он чем-то напоминает массив: состоит из элементов и имеет строго определенную длину. Элементами могут быть любые значения - простые константы или объекты. В отличие от массива, элементы тьюпла не обязательно однородны. А тем, что отличает тьюпл от списка (list) является то, что тьюпл не может быть изменен, т.е. мы не можем i-тому элементу тьюпла присвоить что-то новое и не можем добавлять новые элементы. Таким образом, тьюпл можно назвать списком-константой. Синтаксически тьюпл задается путем перечисления через запятую всех элементов, и все это заключено в круглые скобки:

```
(1,2,5,8)
(3.14,‘string’,-4)
```

Все элементы индексируются с нуля. Для получения i-го элемента необходимо указать имя тьюпла затем индекс i в квадратных скобках. Пример:

```
t=(0,1,2,3,4)

    print t[0], t[-1], t[-3]
Результат: 0 4 2
```

Таким образом, тьюпл можно было назвать вектором-константой, если бы его элементы всегда были однородными.

b. Список. Хорошим, частным примером списка может служить строка (string) языка Turbo Pascal. Элементами строки являются одиночные символы, ее длина не фиксирована, имеется возможность удалять элементы или, напротив, вставлять их в любом месте строки. Элементами же списка могут быть произвольные объекты не обязательно одного и того же типа. Чтобы создать список, достаточно перечислить его элементы через запятую, заключив все это в квадратные скобки:

```
[3, 5.14, ‘s’]
[‘string’, (0,1,8), [1,1]]
```

В отличие от тьюпла, списки можно модифицировать по своему желанию. Доступ к элементам осуществляется также как и в тьюплах. Пример:

```
l = [1, ‘s’, (2,8), [0,3,4]]
print l[0], l[1], l[-2], l[-1][0]
Результат: 1 s (2,8) 0
```

с. Словарь. Напоминает тип запись (record) в Pascal или структуры (structure) в C. Однако, вместо схемы "поле записи"-"значение" здесь применяется "ключ"-"значение". Словарь представляет собой набор пар "ключ"-"значение". Здесь "ключ" - константа любого типа (но преимущественно применяются строки), он служит для именования (индексирования) некоторого соответствующего ему значения (которое можно менять).

Словарь создается путем перечисления его элементов (пар "ключ"-"значение", разделенных двоеточием), через запятую и заключения всего этого в фигурные скобки. Для получения доступа к некоторому значению необходимо, после имени словаря, в квадратных скобках записать соответствующий ключ. Пример:

```
d = {'a': 1, 'b': 3, 5: 3.14, 'name': 'John'}
d['b'] = d[5]
print d['a'], d['b'], d[5], d['name']
Результат: 1 3.14 3.14 John
```

Для добавления новой пары "ключ"-"значение" достаточно присвоить элементу с новым ключом соответствующее значение:

```
d['new'] = 'new value'
print d
Результат: {'a':1, 'b':3, 5:3.14, 'name':'John', 'new':'new value'}
```

3. Python в отличие от Pascal, C, C++ не поддерживает работу с указателями, динамической памятью и адресную арифметику. В этом он похож на Java. Как известно, указатели служат источником трудноуловимых ошибок и работа с ними относится больше к программированию на низком уровне. Для обеспечения большей надежности и простоты они небыли включены в Python.

4. Одним из особенностей Python является то, как происходит присваивание одной переменной другой, т.е. когда по обе стороны от оператора "=" стоят переменные.

Весьма оригинальным является то, как в Python группируются операторы. В Pascal для этого служат операторные скобки *begin-end*, в C, C++, Java - фигурные скобки {}, в Basic применяются закрывающие окончания конструкций языка (NEXT, WEND, END IF, END SUB).В языке Python все гораздо проще: выделение блока операторов осуществляется путем сдвига выделяемой группы на один или более пробелов или символов табуляции вправо относительно заголовка конструкции к которой и будет относиться данный блок. Например:

```
if x > 0:
        print ‘ x > 0 ’
        x = x - 8
else:
        print ‘ x <= 0 ’
        x = 0
```

Тем самым, хороший стиль записи программ.

Единственный недостаток, который удалось обнаружить, - это скорость выполнения программ, которая не всегда может быть такой же высокой, как у программ, написанных на компилирующих языках программирования, таких как C или C++.

Заметить, что в современной реализации Python компилирует (то есть транслирует) инструкции исходного программного кода в промежуточное представление, известное как байт-код, и затем интерпретирует этот байт-код.

Байт-код обеспечивает переносимость программ, поскольку это платформы зависимый формат. Однако из-за того что Python не создает двоичный машинный код (например, машинные инструкции для микропроцессора Intel), некоторые программы на языке Python могут работать медленнее своих аналогов, написанных на компилирующих языках, таких как C.

1.3. Инструменты Python для создания программ

Для создания программы были использованы следующие инструменты:

1. SublimeText 3 – профессиональный текстовый редактор кода, был использован для написания скриптов Python.
2. MicrosoftVisualStudio 2017 CommunityEdition – интегрированная среда разработки, она была использована для создания графического интерфейса пользователя, обработки действий пользователя внутри программы и для создания связи с интерпретатором Python.

2. РАЗРАБОТКА ПРОГРАММЫ «PHYSIC CALCULATION»

2.1. Разработка схемы и функций программы

С помощью модульности Python, ко мне пришла идея создать библиотеку для языка Python с функциями физических уравнений. Схематично это можно представить, как показано на рисунке 3.

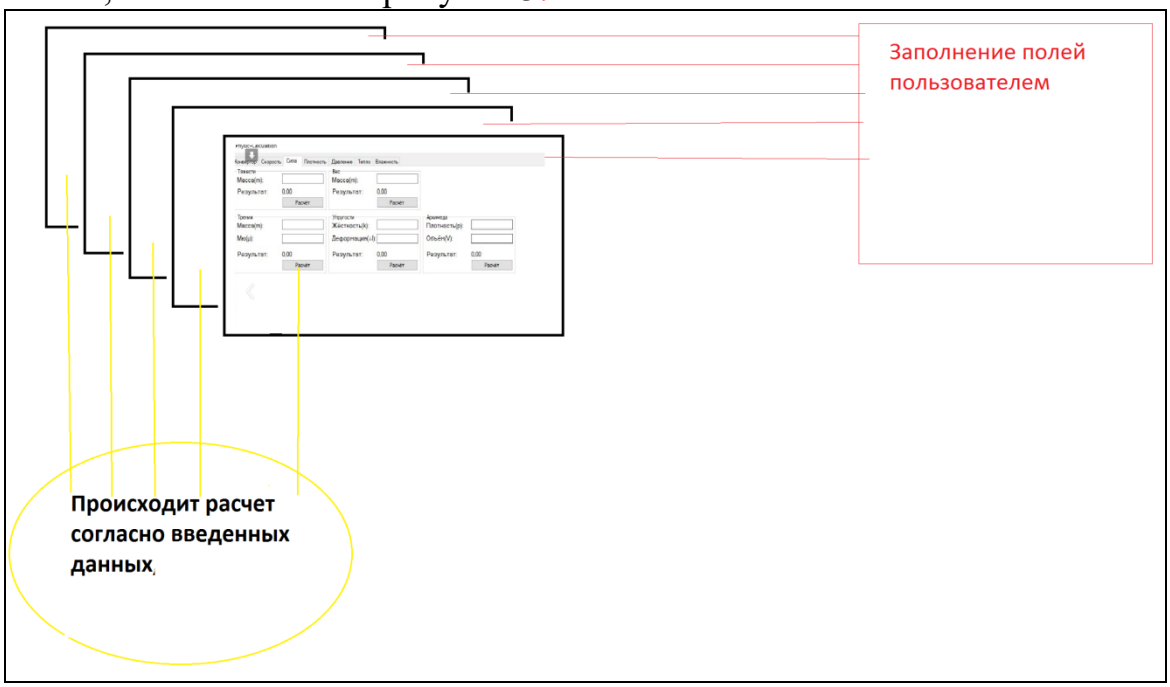


Рис. 3

Прежде чем начать программировать, необходимо было разработать структуру программы, а именно решить где и какая информация будет находиться. Было решено использовать объектно-ориентированное программирование и сделать в каждом файле по классу, в этих классах находились переменные и функции. Физические уравнения, были сделаны в виде методов класса. Всего в модуле 12 файлов.

Результатом данной деятельности стала разработка схемы программы:

```
int/
   modules/
            __init__.py
amperage.py
convert.py
density.py
energy.py
force.py
heat.py
humdity.py
            mechanical.py
            pressure.py
            settings.py
```

Модуль __init__.py (Рис.4) нужен для инициализации программы. Его задача в том, чтобы определить операционную систему компьютера, подключить модуль настроек, импортировать все остальные модули, создать объекты для вызова функций и создать укороченные ссылки на методы классов.

```python
import sys
import math
from os import name, system
path = "_"
if name == "nt":
    path = "\\modules"
elif name == "posix" or "mac":
    path = "/modules"
sys.path.append(sys.path[0]+path)

import settings as sett

from speed      import *
from force      import *
from pressure   import *
from density    import *
from mechanical import *
from energy     import *
from convert    import *
from heat       import *
from humidity   import *
from amperage   import *


def round(numObj, point=sett.point):
    return f"{numObj:.{point}f}"

speed      = speed()
force      = force()
pressure   = pressure()
density    = density()
mechanical = mechanical()
energy     = energy()
convert    = convert()
heat       = heat()
humidity   = humidity()

```

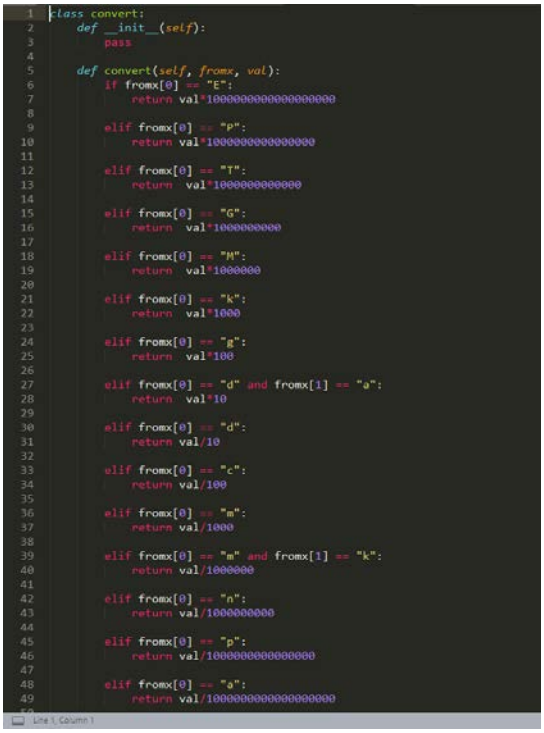
Рисунок 4

Модуль amperage.py (Рис.5) создаёт класс с методами для расчётов электрического тока.

```
class amperage:

    def power(self, q, t):
        return q/t

    def chain(self, u, r):
        return u/r
```

Рисунок 5

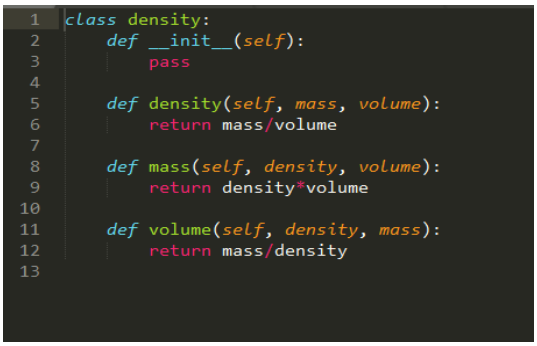
Модуль convert.py (Рис.6) создаёт класс с методами для конвертации многих физических и математических единиц.



```
class convert:
    def __init__(self):
        pass

    def convert(self, fromx, val):
        if fromx[0] == "E":
            return val*1000000000000000000

        elif fromx[0] == "P":
            return val*1000000000000000

        elif fromx[0] == "T":
            return  val*1000000000000

        elif fromx[0] == "G":
            return  val*1000000000

        elif fromx[0] == "M":
            return  val*1000000

        elif fromx[0] == "k":
            return  val*1000

        elif fromx[0] == "g":
            return  val*100

        elif fromx[0] == "d" and fromx[1] == "a":
            return  val*10

        elif fromx[0] == "d":
            return val/10

        elif fromx[0] == "c":
            return val/100

        elif fromx[0] == "m":
            return val/1000

        elif fromx[0] == "m" and fromx[1] == "k":
            return val/1000000

        elif fromx[0] == "n":
            return val/1000000000

        elif fromx[0] == "p":
            return val/1000000000000

        elif fromx[0] == "a":
            return val/1000000000000000
```

Рисунок 6

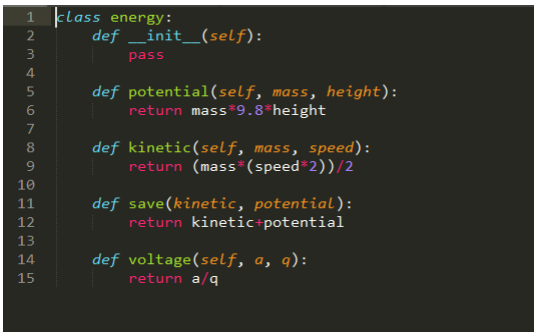
Модуль density.py (Рис.7) создаёт класс с методами для расчётов физической плотности.



```
class density:
    def __init__(self):
        pass

    def density(self, mass, volume):
        return mass/volume

    def mass(self, density, volume):
        return density*volume

    def volume(self, density, mass):
        return mass/density
```

Рисунок 7

Модуль energy.py (Рис.8) создаёт класс с методами для расчётов физической энергии.



```python
class energy:
    def __init__(self):
        pass

    def potential(self, mass, height):
        return mass*9.8*height

    def kinetic(self, mass, speed):
        return (mass*(speed*2))/2

    def save(kinetic, potential):
        return kinetic+potential

    def voltage(self, a, q):
        return a/q
```

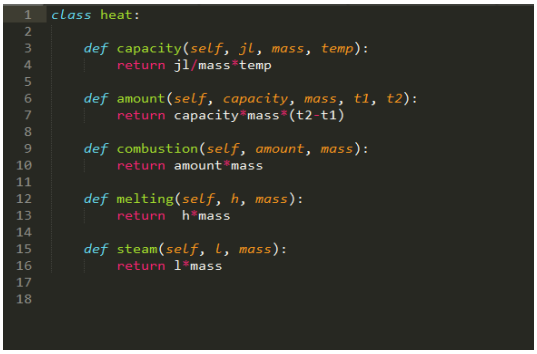
Рисунок 8

Модуль force.py (Рис.9) создаёт класс с методами для расчётов физической силы.

```python
class force:

    g = 9.8

    def __init__(self):
        pass

    def gravity(self, mass):
        return mass*self.g

    def friction(self, mass, mu):
        return mu*mass*self.g

    def elasticity(self, k, l):
        return k*l

    def weight(self, mass):
        return mass*self.g

    def archimedes(self, density_liquid, volume):
        return density_liquid*self.g*volume
```

Рисунок 9

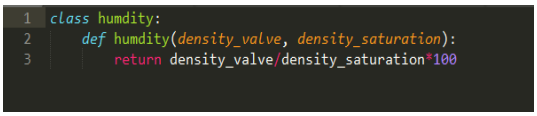
Модуль heat.py (Рис.10) создаёт класс с методами для расчётов физической теплопроводности.



```python
class heat:

    def capacity(self, jl, mass, temp):
        return jl/mass*temp

    def amount(self, capacity, mass, t1, t2):
        return capacity*mass*(t2-t1)

    def combustion(self, amount, mass):
        return amount*mass

    def melting(self, h, mass):
        return  h*mass

    def steam(self, l, mass):
        return l*mass
```

Рисунок 10

Модуль humidity.py (Рис.11) создаёт класс с методами для расчётов физической влажности.



```python
class humdity:
    def humdity(density_valve, density_saturation):
        return density_valve/density_saturation*100
```

Рисунок 11

Модуль mechanical.py (Рис.12) создаёт класс с методами для расчётов физической механики.

```python
class mechanical:
    def __init__(self):
        pass

    def mechanical(self, force, distance):
        return force*distance

    def power(self, mechanical, time):
        return mechanical/time

```

Рисунок 12

Модуль pressure.py (Рис.13) создаёт класс с методами для расчётов физического давления.

```python
class pressure:
    def __init__(self):
        pass

    def pressure(self, force, area):
        return force/area

    def liquid(self, pressure_air, pressure_water):
        return pressure_air-pressure_water

```

Рисунок 13

Модуль settings.py указывает на переменные для настройки программы, пока там есть только переменная для настройки функции округления.

Модуль speed.py (Рис.14) создаёт класс с методами для расчётов скорости, время и расстояния.

```python
class speed:

    def __init__(self):
        pass

    def speed(self, time, distance):
        return distance/time

    def time(self, speed, distance):
        return distance/speed

    def distance(self, speed, time):
        return speed*time

```

Рисунок 14

2.2 Разработка дизайна программы

В современном мире миллиарды вычислительных устройств. Еще больше программ для них. И у каждой свой интерфейс, являющийся «рычагами» взаимодействия между пользователем и машинным кодом. Не удивительно, что чем лучше интерфейс, тем эффективнее взаимодействие.

Для создания графического интерфейса пользователя мной был использован объектно-ориентированный язык программирования C#, в особенности визуальный редактор графического интерфейса пользователя в среде разработки MicrosoftVisualStudio (Рис.15).

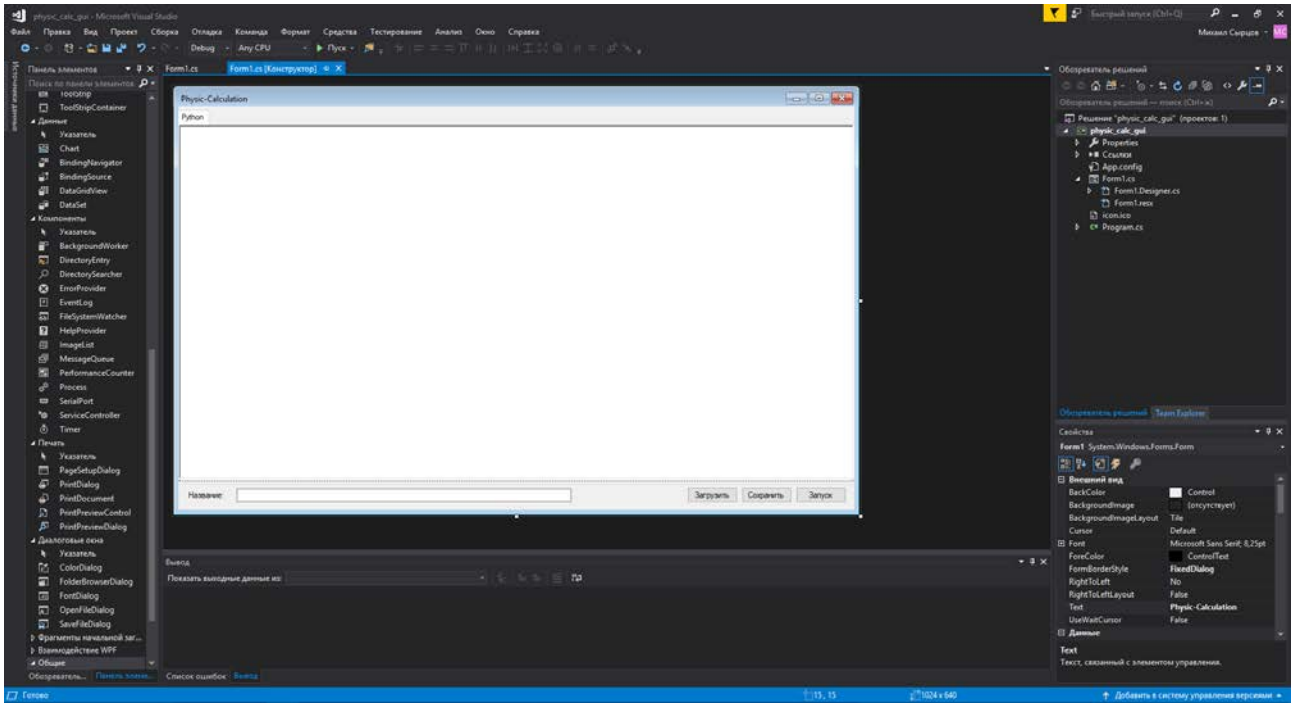


Рисунок 15

Передо мной стояла задача создания удобного и понятного графического интерфейса пользователя. Для этого необходимо было учесть несколько принципов создания графического интерфейса:

- Интерфейс должен быть интуитивно понятным. Таким, чтобы пользователю не требовалось объяснять, как им пользоваться
- Чем быстрее человек увидит результат - тем лучше
- Следует с осторожностью предоставлять пользователю возможность, по установке личных настроек

Для разработки дизайна программы были так же использованы некоторые общепринятые принципы:

- Иконки в программе должны быть очевидными
- Обычно(но не обязательно), элементы размещаются в следующей градации: слева направо, сверху вниз
- Необходимо учитывать привычки пользователя. Например, если в Windows кнопка закрыть находится в правом верхнем углу, то программе аналогичную кнопку необходимо расположить там же. Т.е. интерфейс должен иметь как можно больше аналогий, с известными пользователю вещами
- Соблюдение пропорции
- Если вы даете пользователю информацию, которую он должен куда-то ввести или как-то обработать, то информация должна оставаться на экране до того момента, пока человек ее не обработает

Учитывая данные принципы был разработан следующий графический интерфейс пользователя (Рис. 16).

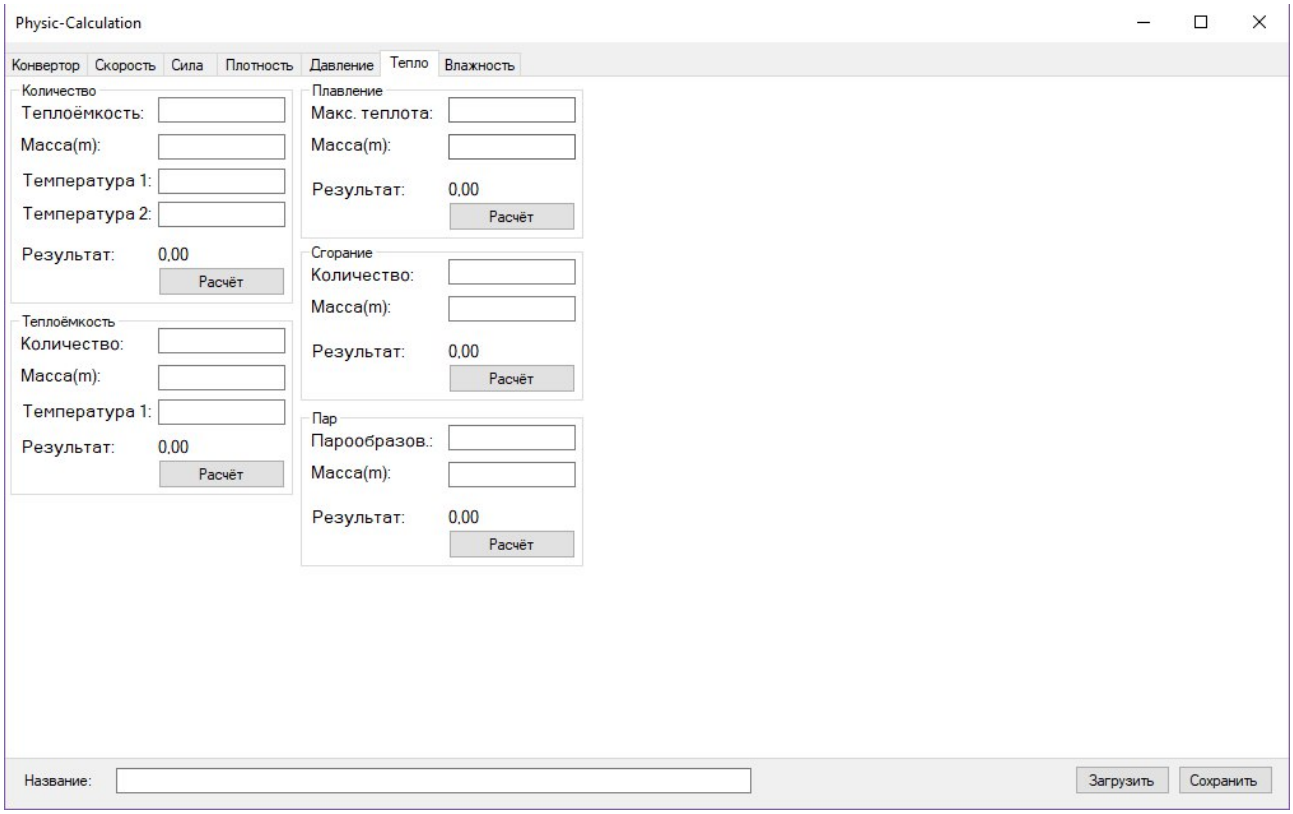


Рисунок 16

В результате расчет со стороны пользователя выглядит так, как показано на рисунке 17

Рис. 17.

Вначале пользователь вводит данные согласно условиям задачи.

Далее нажимает на кнопку расчёта и получает расчет.

Возможность менять параметры, создает условия для анализа и интерпретации результатов, согласно поставленной задаче.

ЗАКЛЮЧЕНИЕ

В результате этой работы, была создана программа для физических расчётов - “Physic-Calculation”. Много правок и кардинальных изменений были внесены в программу, перед тем, как программа обрела свой текущий вид. Скорее всего, и это не финальный вид программы, она будет ещё менять свой внешний вид и возможно будет создана версия на мобильных телефонах.

Программирование в наше время получило большую распространённость, благодаря этому каждый может создать свой сайт или программу, но из-за этого появилось тонна некачественных с функциональной точки зрения или с точки зрения дизайна программ. В своей работе я поставил цель: создать альтернативную программу, которая в себе объединит красивый и понятный дизайн и высокую функциональность.

В первую очередь необходимо было проанализировать существующие программы расчёта физических вычислений и решения задач. Анализ показал, что качественных с точки зрения дизайна и функционала программ почти, что нет.

После этого я начал работу по созданию программы, поскольку я уже знал основы языка Python, мне нужно было только освежить память. Сначала была сделана библиотека для физических расчётов, а затем уже был создан графический интерфейс.

В работе адаптации дизайна была создана программа с текстовым редактором и кнопка вызова интерпретатора Python. После опроса о внешним видом программы, я понял, что людям было бы удобнее вписывать данные в программу и видеть сразу результат, это привело меня к текущему дизайну.

В программе, я так же оставил и возможность писать код самому, скриптами. Это создаёт поддержку программ на Linux и Mac OS.

Эта работа оказалась для меня очень интересной, тем, как можно было реализовать программу. Так же интересна была работа над графическим интерфейсом пользователя.

ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Перышкин. А. В. Физика 7 класс: учебник для общеобразовательных учреждений / А. В. Перышкин. – 15-е изд., стереотип. – М. : Дрофа, 2012. – 191 с.
2. Перышкин. А. В. Физика 8 класс: учебник для общеобразовательных учреждений / А. В. Перышкин. – 15-е изд., стереотип. – М. : Дрофа, 2012. – 191 с.
3. Прохоренок. Н. А. Python 3 и PyQt 5. Разработка приложений / Н.А. Прохоренок, В. А. Дронов. – СПб. : БХВ-Петербург, 2017. – 832 с.
4. Введение в Python. Режим доступаhttps://metanit.com/python/tutorial/1.1.php
5. Краткий обзор языка Python. Режим доступаhttp://www.helloworld.ru/texts/comp/lang/python/python2/
6. Язык программирования: Python и история его создания. Режим доступаhttp://ojde.biz/yazyki-programmirovaniya-python-i-istoriya-ego-sozdaniya/