**III Всероссийская конференция «Юные техники и изобретатели»**

Разработка программного продукта для расчета параметров модельных термодинамических циклов

Номинация «Информационные технологии»

|  |  |
| --- | --- |
|  | Автор: Касумов Магомед Исламович,10 «Г», МБОУ Гимназия,г. Новый Уренгой,Научный руководитель: Атепалихин Михаил Сергеевич, к. п. н., учитель информатики и ИКТ,МБОУ Гимназия, г. Новый Уренгой  |

2016 год

**Оглавление**

[Аннотация 3](#_Toc447529456)

[Введение 4](#_Toc447529457)

[1 Теоретическая часть 6](#_Toc447529458)

[1.1 Физика термодинамического цикла 6](#_Toc447529459)

[1.2 Технологии разработки программного обеспечения, используемые в проекте 9](#_Toc447529460)

[2 Практическая часть 10](#_Toc447529461)

[2.1 Структура программного продукта 10](#_Toc447529462)

[2.2 Перспективы использования программного продукта 13](#_Toc447529463)

[Заключение 13](#_Toc447529464)

[Список литературы 14](#_Toc447529465)

Приложения

Аннотация

В работе описывается проект, направленный на разработку программного продукта, предназначенного для расчета параметров модельных термодинамических циклов и представления результатов расчетов в наглядной форме. Программа позволяет рассчитывать параметры циклов, содержащих изопроцессы, адиабатные процессы и линейные процессы с одноатомным идеальным газом.

В теоретической части работы описаны физические закономерности газовых процессов. Также кратко характеризуются использованные при разработке программы технологии программирования. Практическая часть содержит описание структуры программного продукта и возможные перспективы его использования.

В итоге работы в рамках представляемого проекта был создан программный продукт, автоматизирующий процесс расчета параметров модельных термодинамических циклов. При разработке продукта были использованы современные технологии программирования (объектно-ориентированное программирование, событийно-ориентированное программирование, реализация графического интерфейса пользователя), а также материал углубленного курса физики (работа газа в изотермическом процессе, параметры линейных процессов).

Работа имеет объем 20 машинописных страниц 4 иллюстрации. Список использованных источников включает 5 позиций. В работе имеется 2 приложения с исходным кодом компонентов программного продукта.

Введение

В школьном курсе физики встречаются задачи, требующие достаточно сложных действий для их решения. Они, как правило, подразумевают знание нескольких смежных (а иногда и совсем разных) тем курса, а также выполнения множества алгебраических преобразований и подсчетов. Решение одной такой задачи может отнимать значительное количество учебного времени.

Примером такой задачи является задача расчета параметров термодинамического цикла. Как правило, формулировка такого рода задач подразумевает расчет как параметров отдельных процессов в составе циклического процесса (работа газа на отдельном процессе, количество теплоты, получаемое или отдаваемое газом, изменение внутренней энергии газа и его температуры), так и цикла в целом (работа газа за один цикл, количество теплоты, полученное газом от нагревателя, количество теплоты, отданное газом холодильнику, коэффициент полезного действия цикла). Разбор на уроке задач такого типа может отнимать достаточно много времени. Тем не менее, подобного рода задачи имеют большой потенциал в качестве иллюстрации применения знаний и умений, полученных при изучении таких тем, как «Первый закон термодинамики», «Применение первого закона термодинамики к изопроцессам», «Тепловые двигатели».

Представляется интересной возможность автоматизации решения задач на расчет параметров термодинамических циклов с помощью компьютера. Соответствующая программа может быть использована на уроках физики в качестве демонстрационного пособия при разборе вышеупомянутых задач. Кроме того, программа стала бы удобным инструментом для учителей физики, позволяя формулировать такие задачи для самостоятельных и контрольных работ по термодинамике с готовым решением, что сильно облегчит проверку решений обучающихся.

Кроме того, данный проект позволит авторам освоить прикладное программирование на языке Python, который изучается на уроках информатики, на более высоком уровне. Кроме того, работа позволит углубленно разобраться в отдельных вопросах школьного курса физики.

Итак, **цель проекта** — разработка программы автоматизированного расчета параметров модельных термодинамических циклов, встречающихся в задачах школьного курса физики, на языке программирования Python.

Для достижения поставленной цели необходимо решить ряд **задач**:

1) изучить теорию расчета параметров термодинамических процессов различного типа;

2) изучить новые методики разработки прикладного программного обеспечения: объектно-ориентированное программирование, событийно-ориентированное программирование, разработка программ с графическим интерфейсом пользователя;

3) разработать и реализовать в виде программного модуля алгоритмы расчета физических параметров термодинамических циклов;

4) реализовать программный продукт для расчета параметров термодинамических циклов в виде приложения с графическим интерфейсом пользователя;

5) провести апробацию программного продукта на уроках физики МБОУ Гимназия г. Новый Уренгой.

**Новизна разработки** заключается в создании программного продукта, который позволяет рассчитывать основные параметры модельных термодинамических циклов и наглядно представлять результаты расчетов.

Работа состоит из введения, теоретической части, практической части, заключения и приложений. В теоретической части описывается кратко физическая сторона проекта, а также те технологии программирования, которые были использованы для разработки представляемого программного продукта. Практическая часть посвящена описанию особенностей реализации продукта. В заключении подводятся основные итоги проделанной работы.

1 Теоретическая часть

1.1 Физика термодинамического цикла

Термодинамические циклы — круговые процессы в термодинамике, то есть такие процессы, в которых совпадают начальные и конечные параметры, определяющие состояние рабочего тела: давление, объём, температура. Данные циклы являются моделями процессов, происходящих в реальных тепловых машинах для превращения тепла в механическую работу. Термодинамические циклы состоят из обратимых термодинамических процессов, протекающих последовательно так, что в совокупности составляют замкнутый термодинамический процесс, или цикл[1]. Отдельные процессы цикла сопрягаются между собой узловыми точками.

Расчет параметров термодинамического процессапроизводится с применением первого закона термодинамики:

 $Q=∆U+A$, (1)

где *Q* — количество теплоты, полученное газом, Δ*U* — изменение внутренней энергии газа, *A* — работа, совершаемая газом над внешними телами.

Изменение внутренней энергии идеального одноатомного газа может быть определено по формуле:

 $∆U=\frac{3}{2}νR∆T$, (2)

а с учетом уравнения Клапейрона—Менделеева:

 $pV=νRT ⟹νR∆T=∆\left(pV\right)=p\_{2}V\_{2}-p\_{1}V\_{1}$

 $∆U=p\_{2}V\_{2}-p\_{1}V\_{1}=∆(pV)$. (3)

Здесь и далее используются обозначения: *p* — давление газа, *V* — объем газа, *T* — температура газа, *ν* — количество вещества газа, *R* = 8,31 Дж/(моль · К) — универсальная газовая постоянная.

В задачах, рассматриваемых в рамках данного проекта, термодинамические циклы задаются *pV*—диаграммой. Следовательно, для каждого отдельного процесса всегда имеется возможность вычислить изменение внутренней энергии газа по формуле (3).

Работа газа может быть определена (если иначе затруднительно) из ее геометрического смысла: работа газа при увеличении его объема равна площади под графиком процесса на *pV*—диаграмме.

Количество теплоты, полученное газом может быть вычислено непосредственно по первому закону термодинамики, то есть по формуле (1).

Рассмотрим некоторые достаточно простые термодинамические процессы.

1) ***Изотермический процесс*** — процесс, при котором температура газа остается постоянной.

Изменение внутренней энергии в изотермическом процессе согласно формулы (3) равно нулю. Работу газа можно определить по ее геометрическому смыслу (рис. 1). В курсе математики 11 класса рассматривается способ определения площади под графиком функции $y=f(x)$при заданных пределах изменение переменной *x*. Данный метод подразумевает умение интегрировать функцию. Воспользуемся готовым решением [2]:

 $A=p\_{1}V\_{1}ln\frac{V\_{2}}{V\_{1}}$. (4)



Рис. 1. *pV*—диаграмма изотермического процесса

Количество теплоты, полученное газом в изотермическом процессе, определяем по формуле:

 $Q=A$. (5)

2) ***Изохорный процесс***— процесс, при котором остается постоянным объем газа. В данном процессе работа газа равна нулю ($A=0$). Тогда формула (1) запишется в виде:

 $Q=∆U$. (6)

Изменение внутренней энергии находим по формуле (3).

3) ***Изобарный процесс*** — процесс, в котором не изменяется давление газа. Работа газа в изобарном процессе определяется по изменению его объема:

 $A=p\_{1}(V\_{2}-V\_{1})$. (7)

Изменение внутренней энергии и количество полученной газом теплоты определяем по формулам (3) и (1) соответственно.

4) ***Адиабатный процесс*** — процесс, при котором газ не участвует в теплообмене с окружающими телами ($Q=0$), поэтому совершаемая им работа равна изменению его внутренней энергии, взятому с обратным знаком:

 $A=-∆U$. (8)

Изменение внутренней энергии, как и в предыдущих случаях, определяется по формуле (3).

5) ***Линейный процесс***— процесс, в котором давление изменяется с изменением объема газа линейно (рис. 2).



Рис. 2. *pV*—диаграмма линейного процесса

Процессы данного типа не являются изопроцессами, однако для них достаточно просто находится работа газа по ее геометрическому смыслу: фигура под графиком процесса на *pV*—диаграмме — трапеция, площадь которой находится по известной формуле:

 $A=\frac{1}{2}(p\_{1}+p\_{2})(V\_{2}-V\_{1})$. (9)

Изменение внутренней энергии и количество полученной газом теплоты определяем по формулам (3) и (1) соответственно.

1.2 Технологии разработки программного обеспечения, используемые в проекте

При разработке программного продукта в рамках проекта были использованы следующие технологии программирования:

***1) Создание программ с графическим интерфейсом пользователя.*** Графический интерфейс пользователя ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)graphical user interface,GUI) — разновидность [пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в котором [элементы интерфейса](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0)(меню, кнопки, значки, списки и т.п.), представленные пользователю на [дисплее](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%BB%D0%B5%D0%B9), исполнены в виде [графических](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) изображений. [3]

В данном проекте графический интерфейс пользователя реализуется с помощью библиотеки Tkinter, которая входит в стандартную библиотеку интерпретатора языка Python, является кроссплатформенной библиотекой и может быть использована в большинстве распространённых операционных систем (Windows, Linux, Mac OS X и др.).

***2) Событийно-ориентированное программирование*** — парадигма программирования, в которой выполнение программы определяется событиями — действиями пользователя (клавиатура, мышь), сообщениями других программ и потоков, событиями операционной системы (например, поступлением сетевого пакета). [4]

Событийно-ориентированное программирование можно также определить, как способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

В современных языках программирования события и обработчики событий являются центральным звеном реализации графического интерфейса пользователя. В том числе, данная технология используется и в библиотеке Tkinter.

***3) Объектно-ориентированное программирование*** —это метод программирования, при использовании которого главными элементами программ являются объекты. В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта. Объединение данных и свойственных им процедур обработки в одном объекте, называется инкапсуляцией и является одним из важнейших принципов ООП. [5]

2 Практическая часть

2.1 Структура программного продукта

Программный продукт состоит из следующих компонентов:

***1) модуль tdcycles.py*** с описанием классов для моделирования газа, процесса и цикла. Полный листинг модуля приведен в Приложении 1.

***Класс Gas*** является моделью одноатомного идеального газа и содержит только два атрибута: name (название газа) и amount (количество вещества газа в молях).

***Класс Process*** моделирует газовый процесс. Процесс характеризуется двумя узловыми точками, которые задаются давлением, объемом и температурой. Данный класс имеет атрибуты:

- name — обозначение процесса в задаче (например, «A-B», где A и B — обозначения узловых точек: начальной и конечной);

- type — тип процесса, задаваемый числовой константой (1 — изотермический, 2 — изобарный и т. д.);

- p1, V1, T1, p2, V2, T2 — давление, объем и температура в начальной и конечной точках процесса соответственно;

- gas — экземпляр класса Gas.

Кроме того, в данном классе описывается ряд методов, которые предназначены для расчета параметров процесса:

- dU — возвращает изменение внутренней энергии газа в процессе;

- A — возвращает работу, совершенную газом в процессе;

- Q — возвращает количество теплоты, полученной газом в процессе;

- dT — возвращает изменение температуры газа в процессе.

***Класс Cycle*** моделирует термодинамический цикл и имеет следующие атрибуты:

- gas — экземпляр класса Gas;

- processes — список процессов (экземпляров класса Process), составляющих цикл;

- Q1 — количество теплоты, полученное газом в цикле от нагревателя;

- Q2 — количество теплоты, отданное газом в цикле холодильнику;

- A — полная работа, совершенная газом за цикл.

В классе также реализованы методы:

- addProcess — создает экземпляр класса Process и добавляет его список processes;

- getWorks — возвращает информацию о работе газа на каждом процессе цикла и на цикле в целом в виде текстовой величины;

- getQ — возвращает информацию о количествах теплоты, полученных газом на каждом процессе цикла, а также количества теплоты, полученное от нагревателя и отданное холодильнику в полном цикле, в виде текстовой величины;

- getEF — возвращает коэффициент полезного действия цикла в виде текстовой величины.

***2) модуль td\_cycles\_gui.py*** — графический интерфейс программного продукта. Представляет собою программу—обертку для модуля ***tdcycles***. Полный листинг модуля приведен в Приложении В. Внешний вид приложения, который оно имеет при запуске, приведен на рис. 3.

Программа позволяет последовательно задать информацию о газе (название, количество вещества), добавить нужное количество узловых точек, задать типы процессов в составе цикла и произвести расчет параметров цикла.

При этом, в левой части окна программы выводится *pV*—диаграмма цикла, а в правой части — исходные данные и результаты расчетов (Приложение А, рис. А2).

***3)*** ***файлы ресурсов*** — изображения на кнопках интерфейса.



Кнопка очистки данных цикла (сброс программы в начальное состояние)

Кнопка запуска процесса расчета параметров цикла

Кнопка задания типов процессов в цикле

Кнопка добавления узловой точки цикла

Кнопка задания параметров газа: названия и количества вещества

Рис. 3. Снимок экрана приложения при запуске программы.



Рис. 4. Снимок экрана приложения с результатами расчетов цикла

2.2 Перспективы использования программного продукта

Данная программа может использоваться в разных целях. Например, учитель физики с помощью нее может эффективнее придумывать задачи на расчет параметров термодинамических циклов для самостоятельных и контрольных работ по теме «Основы термодинамики».

Также приложение может быть использовано как иллюстративный материал при закреплении и углублении темы «Тепловые двигатели». Интерфейс программы строился с учетом возможности работы с ней с применением интерактивной доски.

Кроме того, данный программный продукт может быть использован в качестве демонстрационного пособия на уроках информатики профильного (углубленного) уровня. Например, модуль ***tdcycles*** является прекрасной иллюстрацией к теме «Основы объектно-ориентированного обучения». А приложение ***td\_cycles\_gui.py*** позволит продемонстрировать «продвинутые» способы построения графического интерфейса пользователя при изучении соответствующей темы.

Заключение

В итоге работы в рамках представляемого проекта был создан программный продукт, автоматизирующий процесс расчета параметров модельных термодинамических циклов. При разработке продукта были использованы современные технологии программирования (объектно-ориентированное программирование, событийно-ориентированное программирование, реализация графического интерфейса пользователя), а также материал углубленного курса физики (работа газа в изотермическом процессе, параметры линейных процессов).

Дальнейшие перспективные планы по совершенствованию программного продукта:

- доработка графического интерфейса пользователя с целью лучшей адаптации к работе с интерактивной доской;

- реализация функции вывода полных расчетов параметров цикла в формате HTML-документа;

- совершенствование алгоритмов расчета отдельных параметров процессов.

Список литературы

1. Обобщенный цикл тепловых двигателей с газообразным рабочим телом [Электронный ресурс]. [2014]. URL: <http://www.rosavto42.ru/content/termodinamika/teplov_mash.html> (Дата обращения: 15.01.2016).

2. Изотермический процесс [Электронный ресурс] // Молекулярная физика и введение в термодинамику : курс лекций : [сайт]. [2007]. URL: <http://www.formules.ru/material/formula-rabota-gaza-v-izotermicheskom-protsesse/> (Дата обращения: 17.01.2016).

3. Графический интерфейс пользователя [Электронный ресурс] // Википедия: Свободная энциклопедия : [сайт]. [2015]. URL: [https://ru.wikipedia.org/wiki/](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)Графический\_интерфейс\_пользователя. (Дата обращения: 16.01.2016).

4. Событийно-ориентированное программирование. URL: [http://bourabai.ru/alg/ sop.htm](http://bourabai.ru/alg/%20sop.htm)

5. Объектно-ориентированное программирование  URL: [http://bourabai.ru/alg/classifica-tion02.htm](http://bourabai.ru/alg/%20classification02.htm)

**Приложение 1**

from math import log

# types of processes

ISOTHERMAL = 1

ISOBARIC = 2

ISOCHORIC = 3

ADIABATIC = 4

LINEAR = 5

R = 8.31 # J/(mol \* K) -- universal gas constant

def getAmount(p, V, T): return p \* V / R / T

class Gas:

 def \_\_init\_\_(self, name, amount):

 self.name = name

 self.amount = amount

class Process:

 def \_\_init\_\_(self, name, type, p1, V1, p2, V2,

 T1=None, T2=None, gas=None):

 self.name = name

 self.type = type

 self.p1 = p1

 self.V1 = V1

 self.T1 = T1

 self.p2 = p2

 self.V2 = V2

 self.T2 = T2

 self.gas = gas

 if T1 is None: self.T1 = self.p1 \* self.V1 / self.gas.amount / R

 if T2 is None: self.T2 = self.p2 \* self.V2 / self.gas.amount / R

 @property

 def dU(self): return 3 / 2 \* (self.p2 \* self.V2 - self.p1 \* self.V1)

 @property

 def A(self):

 if self.type == ISOCHORIC: return 0

 elif self.type == ISOBARIC: return self.p1 \* (self.V2 - self.V1)

 elif self.type == ISOTHERMAL:

 return self.gas.amount \* R \* self.T1 \* log(self.V2 / self.V1)

 elif self.type == ADIABATIC: return -self.dU

 elif self.type == LINEAR:

 return (self.p1 + self.p2)\*(self.V2-self.V1) / 2

 else: return None

 @property

 def Q(self):

 if self.type == ADIABATIC: return 0

 elif self.type == ISOTHERMAL: return self.A

 elif self.type == ISOBARIC: return 2.5 \* self.A

 elif self.type == ISOCHORIC: return self.dU

 elif self.type == LINEAR: return self.A + self.dU

 else: return None

 @property

 def dT(self):

 return self.T2 - self.T1

class Cycle:

 def \_\_init\_\_(self, gas):

 self.gas = gas

 self.processes = []

 self.Q1 = 0

 self.Q2 = 0

 self.A = 0

 def addProcess(self, name, type, p1, V1, p2, V2):

 proc = Process(name, type, p1, V1, p2, V2, gas=self.gas)

 self.processes.append(proc)

 @property

 def processList(self): return self.processes

 def getWorks(self):

 A = 0

 res = ''

 for p in self.processes:

 res += 'Работа на участке (%s) A = %1.2f Дж\n' % (p.name, p.A)

 A += p.A

 res += 'Полная работа A = %1.2f Дж\n' % A

 self.A = A

 return res

 def getQ(self):

 Q1 = 0

 Q2 = 0

 res = ''

 for p in self.processes:

 res += ('Количество теплоты на участке (%s) '

 'Q = %1.2f Дж\n' % (p.name, p.Q))

 if p.Q > 0: Q1 += p.Q

 else: Q2 += p.Q

 res += ('Количество теплоты, полученное от нагревателя'

 'Q1 = %1.2f Дж\nКоличество теплоты, отданное'

 'холодильнику Q2 = %1.2f Дж\n' % (Q1, Q2))

 self.Q1 = Q1

 self.Q2 = Q2

 return res

 def getEF(self):

 ef = self.A / self.Q1

 return 'КПД = %1.3f или %1.1f %%' % (ef, ef \* 100)

def \_\_test():

 g = Gas('unknown', 0.3)

 c = Cycle(g)

 c.addProcess('1-3', ISOCHORIC, 1, 2, 6, 2)

 c.addProcess('3-5', ADIABATIC, 6, 2, 1, 5.86)

 c.addProcess('5-1', ISOBARIC, 1, 5.86, 1, 2)

 print(c.getWorks())

 print(c.getQ())

 print(c.getEF())

if \_\_name\_\_ == '\_\_main\_\_': \_\_test()

**Приложение 2**

from tkinter import \*

import tkinter.simpledialog as SD

from tkinter.scrolledtext import ScrolledText

from tdcycles import \*

list\_of\_points = []

FONT = "Arial 18"

# handlers

def gas\_click():

 global cyc

 gas\_name = SD.askstring('Название газа',

 'Введите название газа (например, гелий): ')

 if gas\_name == None or gas\_name == '':

 SD.messagebox.showerror('Ошибка!', 'Не введено название газа! ')

 return

 while True:

 try:

 amount = float(SD.askstring('Количество вещества',

 'Введите количество вещества(моль)'))

 except:

 SD.messagebox.showerror('Ошибка!', 'Не введено количество вещества! ')

 continue

 break

 gas = Gas(gas\_name, amount)

 cyc = Cycle(gas)

 txtOut = ('Исходные данные:\nГаз -- %s\n'

 'Количество вещества = %1.2f моль\n' % (gas\_name, amount))

 txtDisplay.insert(END, txtOut)

 btnAddPV.config(state = NORMAL)

def addPV\_click():

 global list\_of\_points

 while True:

 i = len(list\_of\_points) + 1

 pnt = SD.askstring('Информация о точке',

 'Введите информацию о точке %i в формате: '

 '"Обозначение p V"' % (i))

 pnt = pnt.split()

 try:

 pnt\_name = pnt[0]

 pnt\_p = float(pnt[1])

 pnt\_V = float(pnt[2])

 except:

 SD.messagebox.showerror('Ошибка!', 'Не могу понять, что введено! ')

 continue

 break

 list\_of\_points.append((pnt\_name, pnt\_p, pnt\_V))

 txtOut = ('Состояние %s: p = %1.2f x 10^5 Па; '

 'V = %1.2f x 10^-3 м3\n' % (pnt\_name, pnt\_p, pnt\_V))

 txtDisplay.insert(END, txtOut)

 btnGas.config(state = DISABLED)

 if len(list\_of\_points) > 1:

 btnProc.config(state = NORMAL)

def proc\_click():

 btnAddPV.config(state = DISABLED)

 num\_of\_points = len(list\_of\_points)

 list\_of\_points.append(list\_of\_points[0])

 list\_of\_proc\_types = []

 for i in range(num\_of\_points):

 proc\_name = list\_of\_points[i][0]+ '-' + list\_of\_points[i+1][0]

 p1 = list\_of\_points[i][1] \* 1e5

 p2 = list\_of\_points[i+1][1] \* 1e5

 V1 = list\_of\_points[i][2] \* 1e-3

 V2 = list\_of\_points[i+1][2] \* 1e-3

 while True:

 try:

 proc\_type = int(SD.askstring('Тип процесса',

 """ Выберите тип процесса %s

1 -- изотермический;

2 -- изобарный;

3 -- изохорный;

4 -- адиабатный;

5 -- линейный.""" % proc\_name))

 assert proc\_type in (1, 2, 3, 4, 5)

 except:

 SD.messagebox.showerror('Ошибка!', 'Не могу понять, что выбрано! ')

 continue

 break

 cyc.addProcess(proc\_name, proc\_type, p1, V1, p2, V2)

 list\_of\_proc\_types.append((proc\_type, i, i+1))

 d = ['изотермический', 'изобарный', 'изохорный', 'адиабатный', 'линейный']

 txtOut = 'Процесс %s -- %s\n' % (proc\_name, d[proc\_type - 1])

 txtDisplay.insert(END, txtOut)

 btnCalc.config(state = NORMAL)

 draw\_cycle(list\_of\_points, list\_of\_proc\_types)

def play\_click():

 res = cyc.getWorks() + cyc.getQ() + cyc.getEF()

 txtDisplay.insert(END, '-' \* 50 + '\n')

 txtDisplay.insert(END, res)

 btnHtml.config(state = NORMAL)

 btnProc.config(state = DISABLED)

 btnCalc.config(state=DISABLED)

def html\_click():

 SD.messagebox.showwarning('В процессе разработки', 'В процессе разработки')

def clear\_click():

 global list\_of\_points, cyc

 list\_of\_points = []

 try: del cyc

 except: return

 cnvDisplay.delete(ALL)

 txtDisplay.delete('1.0', END)

 btnGas.config(state = NORMAL)

 btnCalc.config(state = DISABLED)

 btnAddPV.config(state = DISABLED)

 btnProc.config(state = DISABLED)

def draw\_cycle(lop, lopt):

 def PtoY(p):

 return 450 - 400 / pmax \* p

 def VtoX(V):

 return 50 + 400 / Vmax \* V

 pmax = 0

 Vmax = 0

 for pnt in lop:

 if pnt[1] > pmax : pmax = pnt[1]

 if pnt[2] > Vmax: Vmax = pnt[2]

 pmax += 1

 Vmax += 1

 cnvDisplay.create\_line(50, 50, 50, 450, 450, 450,

 arrow = BOTH, width = 2, fill='gray')

 cnvDisplay.create\_text(50, 450, text='0', anchor=NE, font = FONT)

 cnvDisplay.create\_text(50, 50, text='p, aтм', anchor=SW, font = FONT)

 cnvDisplay.create\_text(450, 450, text='V, л', anchor=NW, font = FONT)

 for tp in lopt:

 p1 = lop[tp[1]][1]

 p2 = lop[tp[2]][1]

 V1 = lop[tp[1]][2]

 V2 = lop[tp[2]][2]

 if tp[0] in (2, 3, 5):

 x1 = VtoX(V1)

 x2 = VtoX(V2)

 y1 = PtoY(p1)

 y2 = PtoY(p2)

 cnvDisplay.create\_line(x1, y1, x2, y2, width=2, fill ='blue')

 elif tp[0] == 1:

 if V1 > V2:

 V1, V2 = V2, V1

 p1, p2 = p2, p1

 dV = (V2 - V1) / 100

 v = V1

 xx = []

 nu = cyc.gas.amount

 R = 8.31

 T = p1 \* V1 / nu / R

 while v <= V2:

 pv = nu \* R \* T / v

 x = VtoX(v)

 y = PtoY(pv)

 xx.append((x, y))

 v += dV

 cnvDisplay.create\_line(\*xx, width=2, fill ='blue')

 elif tp[0] == 4:

 if V1 > V2:

 V1, V2 = V2, V1

 p1, p2 = p2, p1

 dV = (V2 - V1) / 100

 v = V1

 xx = []

 gamma = 5 / 3

 C = p1 \* V1 \*\* gamma

 while v <= V2:

 pv = C / v \*\* gamma

 x = VtoX(v)

 y = PtoY(pv)

 xx.append((x, y))

 v += dV

 cnvDisplay.create\_line(\*xx, width=2, fill ='blue')

 for pnt in list\_of\_points[:-1]:

 nm = pnt[0]

 p = pnt[1]

 V = pnt[2]

 x = VtoX(V)

 y = PtoY(p)

 r = 5

 cnvDisplay.create\_oval(x-r, y-r, x+r, y+r, outline='red', fill='red')

 cnvDisplay.create\_text(x-r, y-r, text=nm, font=FONT, anchor=SE)

 for v in range(1, int(Vmax)):

 x = VtoX(v)

 cnvDisplay.create\_line(x, 450, x, 455, width=2, fill='gray')

 cnvDisplay.create\_text(x, 460, text=str(v), font=FONT, anchor=N)

 for p in range(1, int(pmax)):

 y = PtoY(p)

 cnvDisplay.create\_line(50, y, 45, y, width=2, fill='gray')

 cnvDisplay.create\_text(40, y, text=str(p), font=FONT, anchor=E)

def info\_click():

 HELP = '''

'''

 wndHelp = Toplevel()

 msg = Message(wndHelp, text=HELP, bg='white', font=FONT,)

 msg.pack()

 wndHelp.resizable(False, False)

 wndHelp.focus\_set()

 wndHelp.grab\_set()

 wndHelp.wait\_window()

# init

root = Tk()

root.geometry('1024x768')

root.title('Термодинамические циклы')

root.resizable(False, False)

# widgets

info\_pic = PhotoImage(file='info.png')

play\_pic = PhotoImage(file='play.png')

arrow\_pic =PhotoImage(file='arrow.png')

html\_pic = PhotoImage(file='html.png')

proc\_pic = PhotoImage(file='process.png')

addpv\_pic = PhotoImage(file='addPV.png')

clr\_pic = PhotoImage(file='x.png')

gas\_pic = PhotoImage(file='gas.png')

btnInfo = Button(root, image=info\_pic, command=info\_click, relief=FLAT)

btnInfo.place(x=10, y=10)

btnCalc = Button(root, image=play\_pic, command=play\_click, relief=FLAT, state=DISABLED)

btnCalc.place(x=470, y=10)

btnAddPV = Button(root, image=addpv\_pic, relief=FLAT,

 command = addPV\_click, state=DISABLED)

btnAddPV.place(x=240, y=10)

btnClr = Button(root, image=clr\_pic, command = clear\_click, relief=FLAT)

btnClr.place(x=907, y=10)

btnProc = Button(root, image=proc\_pic, command = proc\_click, relief=FLAT, state=DISABLED)

btnProc.place(x=355, y=10)

btnGas = Button(root, command=gas\_click, image = gas\_pic, relief=FLAT)

btnGas.place(x=125, y=10)

btnHtml = Button(root, image=html\_pic, relief=FLAT, command=html\_click, state = DISABLED)

btnHtml.place(x=907, y=640)

cnvDisplay = Canvas(root, width=500, height=500, bg='white', relief=SOLID, bd=1)

cnvDisplay.place(x=10, y=130)

txtDisplay = ScrolledText(root, width=59, height=31, relief=SOLID)

txtDisplay.place(x=520, y=132)

Label(root, text='Полное решение в браузере', font=('Arial', 24)).place(x=380, y=670)

l = Label(root, image=arrow\_pic).place(x=820, y=665)

# main loop

root.mainloop()