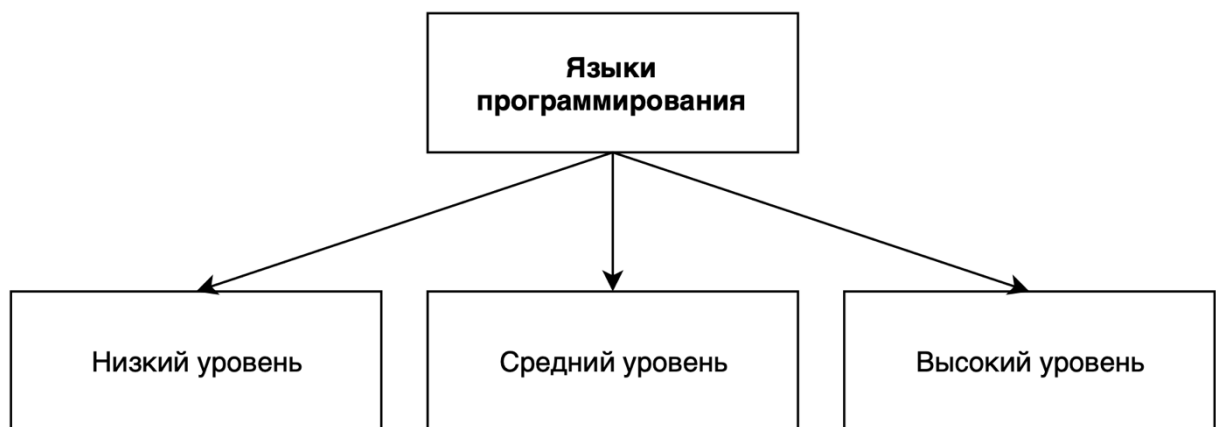


# Технология создания программы "Конвертер величин" на языке программирования Python.

## 1. Выбор языка программирования.

В современном мире существует множество языков программирования, позволяющих разрабатывать программы. Каждый язык обладает своими достоинствами и недостатками. Рассмотрим основные доступные языки программирования и их применимость при разработке программы "Конвертер величин".

Языки программирования подразделяются на низкоуровневые, среднеуровневые и высокоуровневые (рисунок 1).



**Рисунок 1** – Классификация языков программирования по уровням.

*Низкоуровневые языки* программирования близки к программированию непосредственно в машинных кодах. Программист, создающий алгоритм для компьютера на языке низкого уровня, обращаются непосредственно к ресурсам компьютера: процессору, памяти, периферийным устройствам. Это обеспечивает высокую скорость работы программ, поскольку, в отличие от высокоуровневых, в низкоуровневых языках отсутствуют скрытые фрагменты кода, добавляемые автоматически компилятором во время преобразования исходного кода в машинный код. Низкоуровневые языки предполагают, что ответственность за все внутрикомпьютерные ресурсы (время загрузки процессора, объемы выделяемой памяти и т.п.) программист берет на себя. По этой причине языки низкого уровня относятся к "небезопасным": при написании с их помощью более-менее обширных программ, в коде появляется больше ошибок, чем при написании на языках высокого уровня. Низкоуровневое программирование применяется в основном для создания компактного ПО, например, для программирования систем реального

времени (когда недопустимы задержки в работе), встраиваемых систем (микроконтроллеров), а также драйверов, управляющих внешними устройствами:

- принтерами;
- сканерами;
- камерами и т.д.

К плюсам низкоуровневых языков можно отнести:

- полный контроль практически надо всем. Используется только то, что вам нужно;
- большой контроль над памятью. Вы можете сделать то, что практически невозможно в других языках.

Однако, наравне с вышеописанными плюсами, низкоуровневые языки обладают и рядом недостатков:

- дополнительный контроль означает дополнительные сложности, которые могут сделать вроде бы простые задачи более трудными в реализации;
- управление памятью может быстро стать очень сложным.

Среди наиболее популярных низкоуровневых языков выделяют языки C, C++ и Assembler.

*Среднеуровневые языки* программирования служат как бы связующим звеном между аппаратной и программной частью компьютера. Они действуют на уровне абстракции, более понятных для начинающих программистов. Скорость среднеуровневых языков занимает промежуточное положение. Ее не назовешь ни слишком высокой, ни слишком низкой. Среди преимуществ среднеуровневых языков программирования можно выделить:

- управление памятью является необязательным;
- богатые стандартные программные библиотеки;
- возможность писать программы, которые запускаются на любом компьютере и операционной системе.

Среди минусов среднеуровневых языков выделяют:

- несмотря на то, что управление памятью является необязательным, программисту нужно понимать, как устроен процесс управления памятью;
- для запуска программы на любом компьютере и операционной системе пользователю требуется установка специальной среды выполнения;
- хоть язык и обладает большим количеством абстракций, иногда требуется фундаментальное знание в области взаимодействия с внутрикомпьютерными ресурсами.

Наиболее популярные среднеуровневые языки программирования – Java, C#.

*Высокоуровневые языки* программирования рассчитаны на то, чтобы они были наиболее просты в освоении и понятны человеку. Они не зависят от аппаратной части, то есть программы на высокоуровневом языке могут быть запущены на любой операционной системе. Кроме того, высокоуровневые языки самые простые в освоении и позволяют в более сжатые сроки разрабатывать программные продукты. Высокоуровневые языки обладают наибольшим числом абстракций и делают процесс разработки программ наиболее приятным. Среди плюсов высокоуровневых языков выделяют:

- высокий уровень абстракций, что делает сложные задачи простыми;
- исходный код программы прост и интуитивен при чтении;
- огромное количество стандартных и сторонних библиотек, решающих наиболее распространенные задачи при разработке программ;
- независимость от платформы: программы на высокоуровневом языке могут быть запущены на любом компьютере и на любой операционной системе.

Основным недостатком высокоуровневых языков является скорость выполнения программ. Высокий уровень абстракций снижает производительность программ, поэтому в сравнении с низкоуровневыми и среднеуровневыми языками программы, написанные на высокоуровневых языках, являются самыми медленными в исполнении. Среди высокоуровневых языков программирования выделяют: Python, Ruby, JavaScript.

Предполагается, что разрабатываемая программа “Конвертер величин” должна иметь возможность запускаться на любой операционной системе независимо от аппаратной составляющей компьютера пользователя. Из-за этого, для её разработки логично остановиться на среднеуровневых и высокоуровневых языках. Кроме того, так как разработка программы проходит в рамках школьного проекта, стоит остановиться на наиболее простом в изучении и освоении языке программирования. Согласно вышеописанной характеристике, такими языками являются высокоуровневые языки программирования.

Среди высокоуровневых языков программирования наибольшей популярностью при разработке программ обладает язык Python. Python — это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым языком, на Python можно написать практически что угодно без ощутимых проблем. Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нём не писал.

За счёт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее, чем на других вышеописанных языках. Кроме того, язык Python обладает наибольшим количеством библиотек, которые решают наиболее часто встречающиеся проблемы при разработке программного кода.

В итоге, проведя анализ существующих языков программирования, было решено, что для разработки программы “Конвертер величин” будет использоваться язык программирования Python.

## 2. Концепция и возможности программы.

В основе разрабатываемой программы лежит 3 основных понятия: *величина*, *значение величины* и *единица измерения*.

*Величины* используются для описания количественных свойств физических объектов, явлений и процессов. К примеру, в физике величинами являются длина, ширина, высота, масса, объем, скорость, время, ускорение, и др.

Величины всегда имеют какие-то *значения*. Так скорость одного тела может быть 10 км/ч, а скорость другого быть равной 20 км/ч. Здесь мы имеем одну и ту же физическую величину — скорость. Однако ее значения различны.

Значение величины всегда выражается в каких-то конкретных *единицах измерения*. Другими словами, значение величины является именованным числом. Так масса может выражаться в килограммах, граммах, тоннах и др.; время — в часах, минутах, годах и др.

Основной целью программы является упрощение возможности перевода значений между единицами измерения в рамках конкретной величины. При работе с программой пользователь выбирает величину, с которой он хочет работать. Далее, выбирается 2 единицы измерения: исходная и конечная. Исходная единица измерения — та единица измерения, из которой предстоит конвертация. Конечная единица измерения — та единица измерения, в которую происходит конвертация. После этого пользователь вводит значение величины в исходных единицах и после подтверждения программа производит расчет значения величины в конечной единице измерения.

В большинстве аналогов разрабатываемой программы имеется один большой недостаток: в программе присутствует список величин с соответствующими единицами измерения, которые в дальнейшем нельзя изменять. Это не всегда удобно, так как может оказаться, что необходимая пользователю физическая величина просто отсутствует в стандартном наборе. В связи с этим было решено, что разрабатываемая программа будет расширяемая. Простыми словами, в программе будет возможность добавлять

произвольные величины и единицы измерения. Таким образом, даже если пользователь не найдет в стандартном списке нужную величину или единицу измерения, он всегда сможет расширить стандартный набор. Эта возможность делает разрабатываемую программу гораздо полезнее и функциональнее, чем существующие аналоги и конкуренты.

### 3. Дополнительные инструменты, библиотеки и технологии.

Так как разрабатываемая программа является расширяемой, встает вопрос о необходимости сохранения добавляемых пользователем величин или единиц измерения в какой-либо базе данных.

Наиболее простым решением является сохранение новых величин или единиц измерения в дополнительном файле, хранимом на компьютере пользователя. При добавлении новой величины или единицы измерения программа будет записывать новые данные в файл, а при загрузке программы считывать эти данные из файла. Таким образом при перезагрузке программы пользователь не потеряет добавленные величины и единицы измерения и сможет пользоваться ими дальше.

В качестве формата хранения данных был выбран формат JSON. – это общий формат для представления значений и объектов. Его описание задокументировано в стандарте RFC 4627. Первоначально он был создан для JavaScript, но многие другие языки также имеют библиотеки, которые могут работать с ним. Таким образом, JSON легко использовать для обмена данными, сохранения данных при разработке программ на любом современном языке программирования. Пример данных в формате JSON представлен в листинге 1.

```
[
  {
    "name": "Масса",
    "units": [
      {
        "name": "Килограммы",
        "conversion_factor": 1.0,
        "conversion_operation": "*"
      },
      {
        "name": "Граммы",
        "conversion_factor": 0.001,
        "conversion_operation": "*"
      },
      {
        "name": "Тонны",
        "conversion_factor": 1000.0,
        "conversion_operation": "*"
      }
    ]
  }
]
```

**Листинг 1** – Пример данных, записанных в файл в формате JSON.

Для написания пользовательского интерфейса программы существует ряд сторонних библиотек, значительно упрощающих данный процесс. Одной из популярных библиотек является библиотека Tkinter. Библиотека Tkinter содержит набор стандартных компонент графического интерфейса пользователя. Под графическим интерфейсом пользователя подразумеваются все те окна, кнопки, текстовые поля для ввода, списки, флажки и др., которые вы видите на экране, открывая то или иное приложение. Библиотека Tkinter является достаточно простой, при этом она предоставляет все необходимое для разработки пользовательского интерфейса программы. Именно поэтому при разработке программы “Конвертер величин” использовалась именно эта библиотека.

#### 4. Процесс конвертации величин.

Рассмотрим непосредственно процесс конвертации величин в разрабатываемой программе.

Каждая величина, присутствующая в программе, имеет ряд единиц измерения. Среди списка единиц измерения в рамках каждой величины присутствует базовая. Каждая единица измерения имеет 3 основных характеристики: название, фактор преобразования и операция преобразования. Рассмотрим подробнее такие параметры, как фактор преобразования и операция преобразования.

Как правило, при переводе из одной единицы измерения в другую чаще всего возникает необходимость умножения значения величины в исходных единицах на какой-либо коэффициент или прибавление к значению величины в исходных единицах какого-либо коэффициента. Так, для перевода массы из килограммов в граммы значение величины нужно умножить на 1000, а при переводе температуры из градусов Цельсия в Кельвины необходимо к значению величины прибавить 273. Таким образом, умножение или сложение назовем *операцией преобразования*, а коэффициент – *фактором преобразования*.

Для каждой величины выбирается базовая единица измерения, фактор преобразования для которой равен 1. Для всех остальных единиц измерений в рамках одной величины проставляются факторы преобразования и операции преобразования таким образом, чтобы, применив операцию преобразования с соответствующим фактором к выбранной единице измерения, она приобретала эквивалент базовой единицы измерения.

Конвертация значения величины из исходной единицы измерения в конечную выполняется в 2 этапа. На первом этапе происходит преобразование значения из исходной

единицы измерения в базовую с использованием соответствующей операции преобразования по формуле 1:

$$V_{\text{базовая}} = V_{\text{исходная}} * / + F_{\text{исходная}} \quad (1)$$

где:

$V_{\text{базовая}}$  – значение величины в базовой единице измерения;

$V_{\text{исходная}}$  – значение величины в исходной единице измерения;

$F_{\text{исходная}}$  – фактор преобразования исходной единицы измерения.

После перевода значения величины из исходной единицы измерения в базовую становится возможным получить значение величины в конечной единице измерения (формула 2):

$$V_{\text{конечная}} = V_{\text{базовая}} \div / - F_{\text{конечная}} \quad (1)$$

где:

$V_{\text{конечная}}$  – значение величины в конечной единице измерения;

$F_{\text{конечная}}$  – фактор преобразования конечной единицы измерения.

Для ясности работы алгоритма приведем конкретный пример. Пусть у нас есть 3 единицы измерения: граммы, килограммы и тонны в рамках одной величины – масса. Операцией преобразования в данном случае будет являться умножение. Базовой величиной выберем килограммы и установим для неё фактор преобразования, равный 1. Для граммов и тонн факторы преобразования будут 0,001 и 1000 соответственно. Поставим перед собой задачу перевода 1000 грамм в тонны. Согласно формуле 1, на первом этапе переведем граммы в базовую величину - килограммы:

$$V_{\text{базовая}} = V_{\text{исходная}} * F_{\text{исходная}} = 1000 * 0,001 = 1 \text{ (кг)}$$

Далее, согласно формуле 2, переведем полученное значение в базовой единице (килограммы) в конечную единицу (тонны):

$$V_{\text{конечная}} = V_{\text{базовая}} \div F_{\text{конечная}} = 1 \div 1000 = 0,001 \text{ (т)}$$

Итого получаем, что 1000 грамм – это 0,001 тонна, что истинно.

Исходный код функции конвертации на языке программирования Python представлен в листинге 2.

```
def convert(self):
    gotten_value_str = self.first_entry.get()
    first_unit_str = self.selected_unit_one.get()
    second_unit_str = self.selected_unit_two.get()

    if gotten_value_str == " or first_unit_str == " or second_unit_str == ":
        showerror("Некорректные данные", "Заполните все поля для конвертации")
        return

    try:
        gotten_value = float(gotten_value_str)
```

```

except ValueError:
    showerror("Введите корректные данные", "Введите числовое значение")
    return

gotten_first_unit = self.units[first_unit_str]
gotten_second_unit = self.units[second_unit_str]

if gotten_first_unit.conversion_operation == "**":
    first_step_value = gotten_value * gotten_first_unit.conversion_factor
elif gotten_first_unit.conversion_operation == "+":
    first_step_value = gotten_value + gotten_first_unit.conversion_factor

if gotten_second_unit.conversion_operation == "**":
    second_step_value = first_step_value / gotten_second_unit.conversion_factor
elif gotten_second_unit.conversion_operation == "+":
    second_step_value = first_step_value - gotten_second_unit.conversion_factor

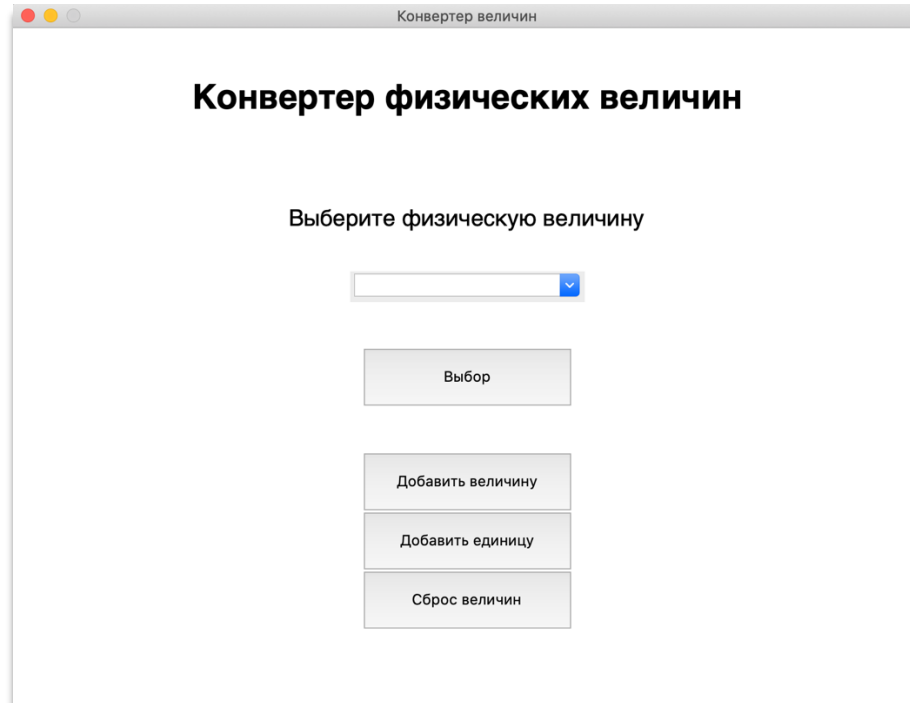
```

**Листинг 2** – Исходный код функции конвертации значения величины между выбранными пользователем единицами измерения.

## 5. Результат разработки программы.

В итоге, согласно требованиям из раздела 2 и алгоритму, описанному в разделе 4, была разработана программа “Конвертер величин”.

Начальный экран программы содержит пункты меню, соответствующие базовым возможностям программы (рисунок 2).

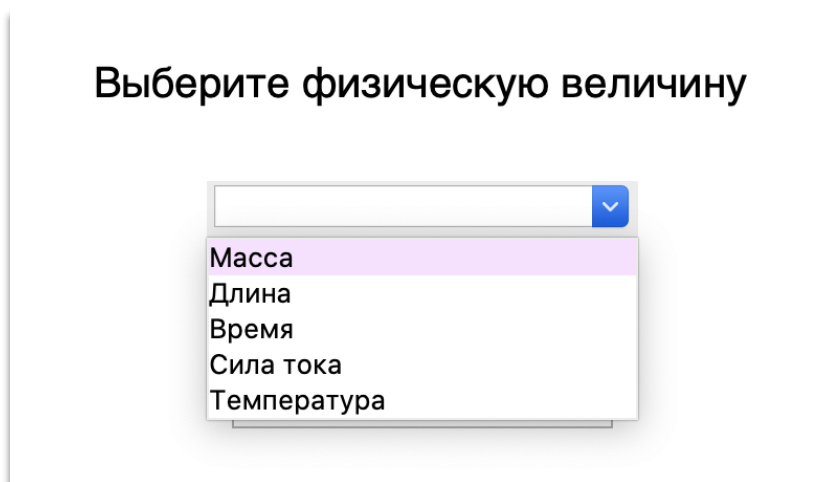


**Рисунок 2** – Начальный экран программы.

Начальный экран содержит окно выбора величины, в рамках которых необходимо провести преобразование, а также пункты меню, отвечающие за добавление новых величин и единиц измерений, а также сброса настроек программы до первоначальных.

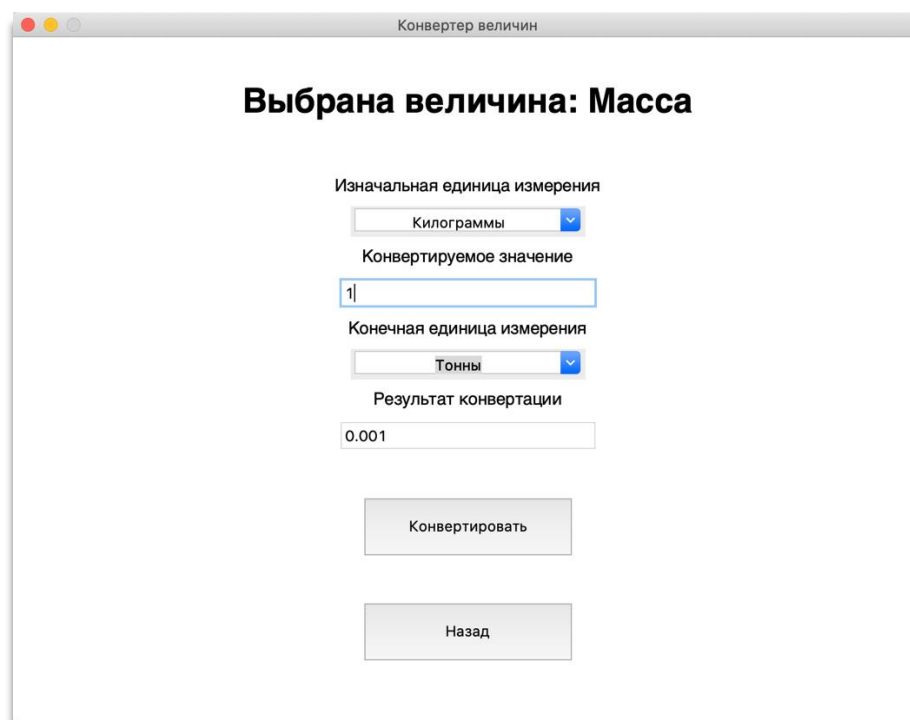


При первом запуске программы в нее заложен базовый набор величин (масса, длина, время, сила тока и температура) с соответствующими единицами измерения (рисунок 3).



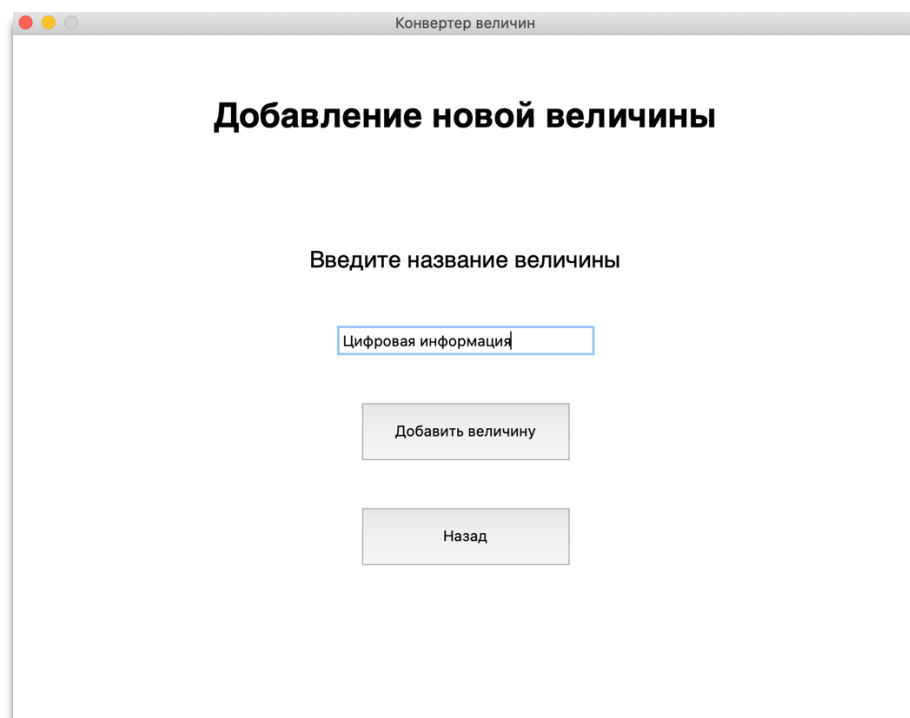
**Рисунок 3** – Базовый набор величин программы.

При выборе величины и нажатии на начальном экране кнопки “Выбор” открывается экран конвертации величин (рисунок 4). Пользователь выбирает изначальную и конечную единицу измерения выбранной величины, а также вводит исходное значение. После нажатия на кнопку “Конвертировать” в поле “Результат конвертации” появляется значение величины в конечной единице измерения.



**Рисунок 4** – Экран конвертации значений величин.

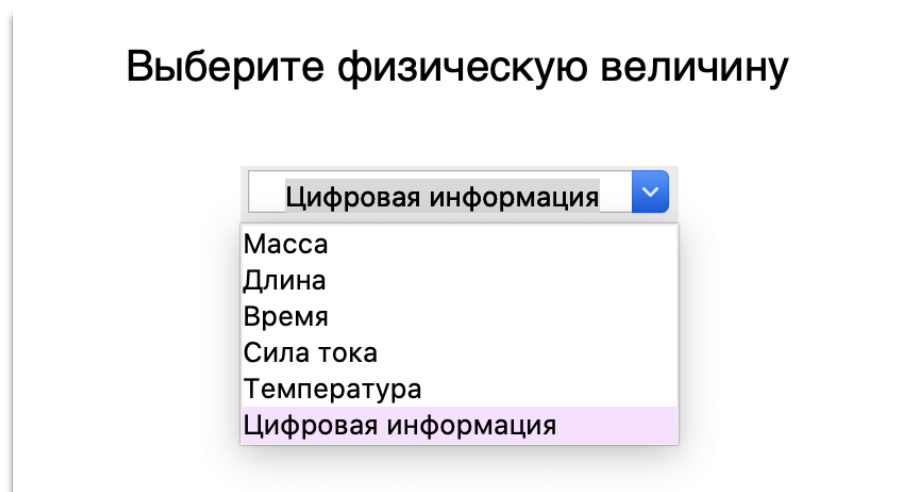
При выборе на начальном экране пункта меню “Добавить величину” открывается экран добавления новой величины (рисунок 5). На этом экране необходимо ввести название новой величины и нажать кнопку “Добавить величину”.



The screenshot shows a window titled "Конвертер величин" (Unit Converter). The main heading is "Добавление новой величины" (Adding a new quantity). Below it, the instruction "Введите название величины" (Enter the name of the quantity) is displayed. A text input field contains the text "Цифровая информация" (Digital information). Below the input field are two buttons: "Добавить величину" (Add quantity) and "Назад" (Back).

**Рисунок 5** – Экран добавления новой величины.

После добавления новой величины она появится в списке на начальном экране (рисунок 6).

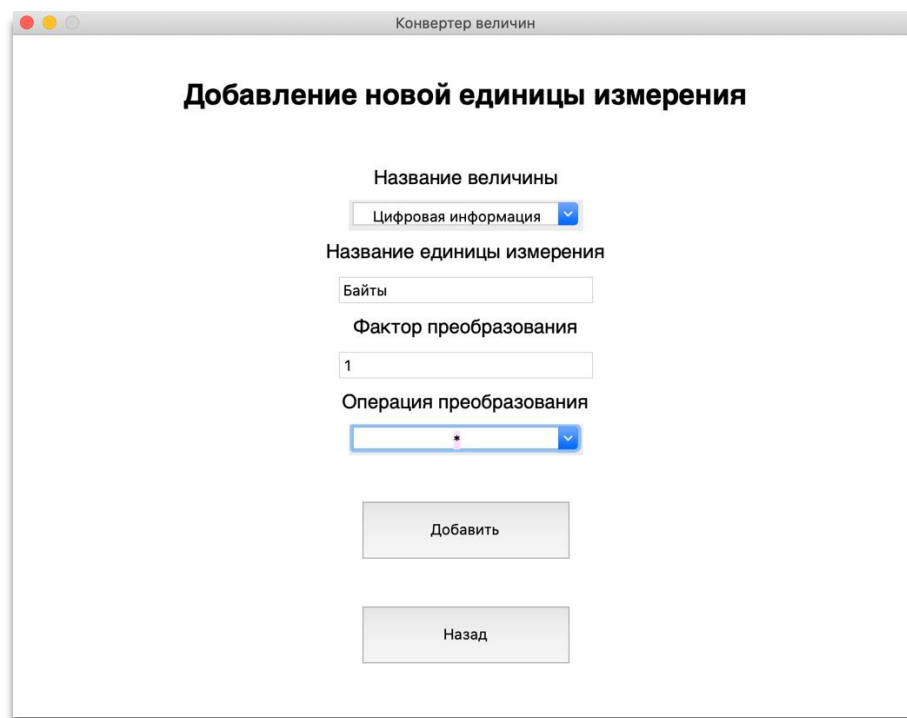


The screenshot shows a window titled "Выберите физическую величину" (Select a physical quantity). It features a dropdown menu with the current selection "Цифровая информация" (Digital information). The dropdown is open, showing a list of options: "Масса" (Mass), "Длина" (Length), "Время" (Time), "Сила тока" (Current), "Температура" (Temperature), and "Цифровая информация" (Digital information), which is highlighted at the bottom.

**Рисунок 6** – Список величин после добавления новой величины.

При нажатии на начальном экране на кнопку “Добавить единицу измерения” открывается экран добавления новой единицы измерения (рисунки 7,8). Пользователю предоставляется выбрать величину, для которой происходит добавление единицы

измерения, а также ввести основные параметры единицы измерения: название, фактор преобразования и операцию преобразования.



Конвертер величин

### Добавление новой единицы измерения

Название величины  
Цифровая информация

Название единицы измерения  
Байты

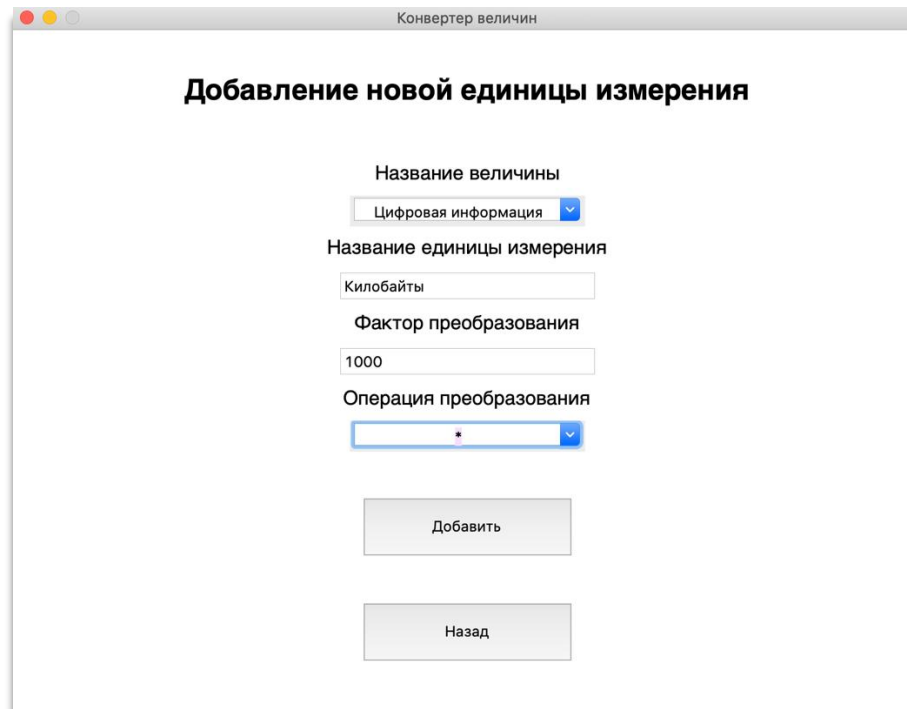
Фактор преобразования  
1

Операция преобразования  
\*

Добавить

Назад

**Рисунок 7** – Добавление единицы измерения “Байты” для величины “Цифровая информация”.



Конвертер величин

### Добавление новой единицы измерения

Название величины  
Цифровая информация

Название единицы измерения  
Килобайты

Фактор преобразования  
1000

Операция преобразования  
\*

Добавить

Назад

**Рисунок 8** – Добавление единицы измерения “Килобайты” для величины “Цифровая информация”.

После добавления величины “Цифровая информация” (рисунок 5) и единиц измерения “Байты” и “Килобайты” (рисунки 7, 8) есть возможность конвертации единиц измерения в рамках этой величины (рисунок 9).

The screenshot shows a window titled "Конвертер величин". The main heading is "Выбрана величина: Цифровая информация". Below this, there are four labels and their corresponding input fields: "Изначальная единица измерения" (Initial unit of measurement) with a dropdown menu showing "Килобайты"; "Конвертируемое значение" (Value to be converted) with a text input field containing "1"; "Конечная единица измерения" (Final unit of measurement) with a dropdown menu showing "Байты"; and "Результат конвертации" (Conversion result) with a text input field showing "1000.0". At the bottom, there are two buttons: "Конвертировать" (Convert) and "Назад" (Back).

**Рисунок 9** – Конвертация значений добавленных единиц измерения в рамках добавленной физической величины.

В заключение, если возникла необходимость в отчистке добавленных пользователем единиц измерения и величин, есть возможность вернуться к базовому списку при помощи кнопки “Сброс величин” на начальном экране (рисунок 10).

The screenshot shows a window titled "Конвертер физических величин". In the center, there is a smaller dialog box titled "Величины сброшены" (Units reset) with the text "Список физических величин сброшен до первоначальных" (The list of physical quantities has been reset to the original ones) and an "OK" button. Below the dialog box, there are four buttons: "Выбор" (Select), "Добавить величину" (Add quantity), "Добавить единицу" (Add unit), and "Сброс величин" (Reset quantities).

**Рисунок 10** – Демонстрация сброса списка величин и единиц измерения до первоначальных.

## 6. Заключение.

В ходе проекта была разработана программа “Конвертер величин” на языке программирования Python.

В ходе работы над проектом были рассмотрены основные современные языки программирования, их достоинства и недостатки. В качестве языка программирования для разработки программы был выбран язык Python из-за его простоты в освоении, высокоуровневых абстракций, большого числа библиотек, предназначенных для решения общеизвестных задач а также из-за возможности запуска программ, разрабатываемых на этом языке, на любой операционной системе и на любом компьютере.

Был разработан алгоритм, позволяющий при помощи основных параметров единиц измерения преобразовывать значения величин из одной единицы измерения в другую.

При помощи современных подходов к хранению данных программа была снабжена функционалом, отличающим её от существующих аналогов, а именно – расширяемостью величин и единиц измерения, доступных пользователю.

В итоге можно сделать вывод о том, что основная концепция и возможности программы, которые предполагались при составлении идеи проекта, были успешно реализованы.