***Лабораторная работа №16***

**Выбор и обоснование средств реализации проекта для разрабатываемой программной системы.**

**Цель работы:**

1. Изучить критерии выбора языка программирования. Научиться обосновывать выбор среды разработки в  соответствии с критериями выбора языка программирования.

2. Изучить основные парадигмы программирования.

3. Изучить правила формирования листинга программы.

**Порядок выполнения работы:**

**1. Выполните задания 1-3.**

**Задание 1. Критерии выбора языка программирования.**

1. Изучить критерии выбора языка программирования. (если информация отсутствует в конспекте, то  необходимо ее законспектировать).

2. Обосновать выбор языка программирования для решения поставленной задачи в соответствии с  индивидуальным заданием.

Для решения поставленной задачи я выбрал C#.

C# — это мультипарадигмальный высокоуровневый язык программирования, поддерживающий несколько парадигм, таких как структурное, императивное, объектно-ориентированное, событийно-ориентированное, функциональное и обобщенное программирование. C# был разработан компанией Microsoft и является одним из основных языков платформы .NET. А также он имеет множество особенностей, которые отличают его от других языков C-подобной семьи. Например, C# поддерживает портативность, сильную типизацию, метапрограммирование, свойства, пространства имен, управляемый доступ к памяти и исключения. C# также поддерживает полиморфизм, интегрированный язык запросов (LINQ) и функциональное программирование.

Для разработки на C# вам понадобятся следующие инструменты:

• Visual Studio Code - это бесплатная и мощная среда разработки с открытым исходным кодом, которая поддерживает множество языков и расширений.

• C# Dev Kit - это расширение для Visual Studio Code, которое предоставляет вам все необходимые функции для работы с C#, такие как подсветка синтаксиса, автодополнение кода, отладка, тестирование и развертывание.

• .NET SDK - это набор инструментов для создания приложений на базе .NET, включающий компиляторы, библиотеки и среду выполнения.

Например, вы можете создавать:

• Десктопные приложения для Windows, используя Windows Forms, WPF или UWP.

• Веб-приложения и веб-сервисы, используя ASP.NET Core, Blazor или SignalR.

• Мобильные приложения для Android, iOS или Windows, используя Xamarin.

• Игры для разных платформ, используя Unity.

• Приложения машинного обучения, используя ML.NET.

• и т.д.

Для разработки на C# существуют различные ресурсы, которые помогут вам в обучении C# такие как:

• Документация по C# и .NET, которую вы можете найти на сайте [learn.microsoft.com] или в [Википедии].

• Обучающие материалы, такие как курсы, книги, видео и статьи, которые помогут вам изучить основы и продвинутые темы C#. Вы можете найти некоторые из них на сайте [learn.microsoft.com] или в [Stack Overflow].

• Библиотеки и фреймворки, которые расширяют функциональность C# и .NET, предоставляя готовые решения для часто встречающихся задач. Вы можете найти их в реестре NuGet или на GitHub.

**Задание 2. Изучение парадигм программирования**

1. Изучить различные парадигмы программирования. (если информация отсутствует в конспекте, то  необходимо ее законспектировать).

*2.* Провести сравнительную характеристику различных парадигм программирования. Результаты  оформить в виде таблицы следующего вида:

Дайте характеристику следующим **парадигмам программирования**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название парадигмы** | **Сущность**  **парадигмы,  основные  идеи,**  **принципы,**  **объекты.** | **Языки**  **поддерживающие данную  парадигму.**  **(2-3 языка)** | **Достоинства** | **Недостатки** | **Примеры**  **программ** |
| **1.** | **Императивное**  **программирование** | Императивная программа похожа на приказы | Ruby, C++ | Простота в понимании  Производительность | Сложность поддержания  Ограничения абстракции | Рисунок 1 |
| **2.** | **Декларативное**  **программирование** | Главное результат | SQL, CSS | Упрощение кода, абстракция,  Повышение производительности | Ограниченная гибкость, сложность понимания | Рисунок 2 |
| **3.** | **Структурное**  **программирование** | Разбиение на блоки | Python, Pascal | Чёткая организация кода, легкость сопровождения, производительность | Трудность поддержания гибкости, сложность работы с большими проектами | Рисунок 3 |
| **4.** | **Процедурное**  **программирование** | Разбиение на функции(процедуры) | Python, С | Модульность, чёткая структура, простота в понимании | Ограниченная гибкость, отсутствие инкапсуляции | Рисунок 4 |
| **5.** | **Модульное**  **программирование** | Разбиение на модули | JS, Ruby | Модульность Изолирование ошибок | Управление интерфейсами, сложность интеграции | Рисунок 5,  Рисунок 6 |
| **6.** | **Объектно**  **ориентированное**  **программирование** | организует программу вокруг объектов, которые представляют собой экземпляры классов | Java, C++, Python | Модульность и переиспользование кода, Инкапсуляция, Наследование, Разработка в команде | Сложность, Избыточность, Производительность | Рисунок 7 |
| **7.** | **Функциональное**  **программирование** | программа рассматривается как набор функций, которые преобразуют данные | Haskell, Clojure,  Scala | Чистота функций, Отсутствие побочных эффектов, Легкость тестирования | Сложность изучения, Производительность, Сложность при работе с вводом/выводом | Рисунок 8 |
| **8.** | **Логическое**  **программирование** | Основана на математической логике | Prolog, Mercury, Python | Декларативность, Легко читаемый код, Простота разработки | Ограниченная эффективность выполнения, Сложность отладки, Понимание сложных программ | Рисунок 9 |

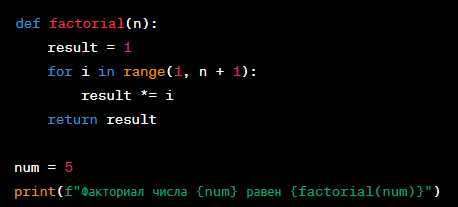


Рисунок 1

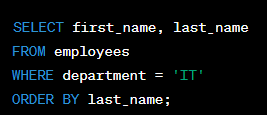


Рисунок 2

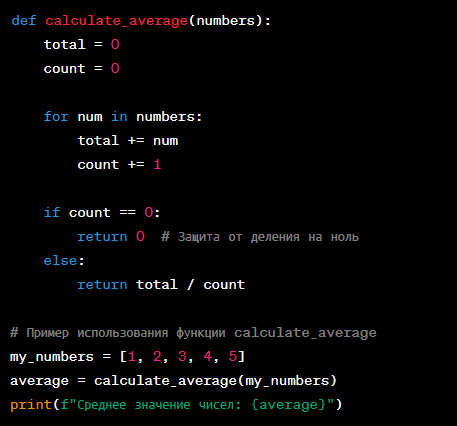


Рисунок 3

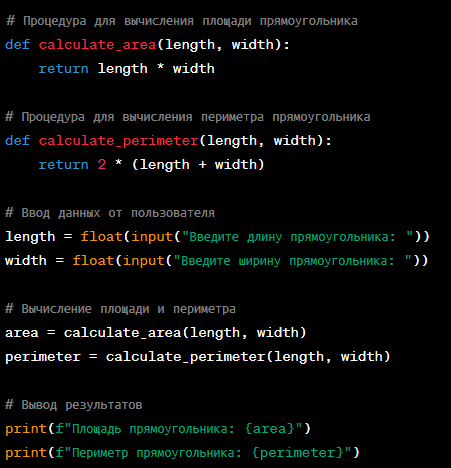


Рисунок 4

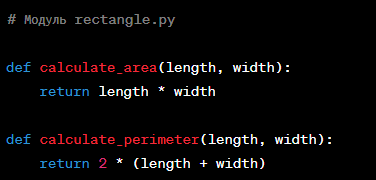


Рисунок 5

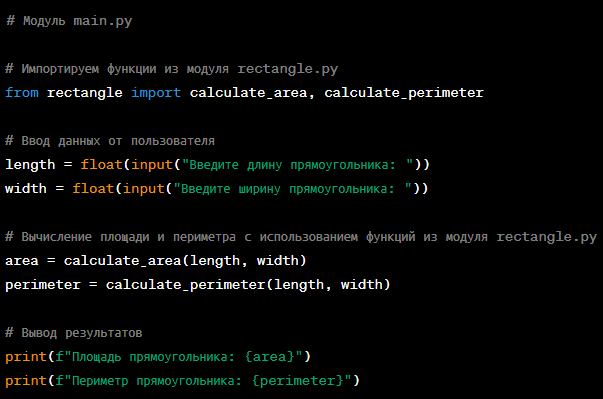


Рисунок 6

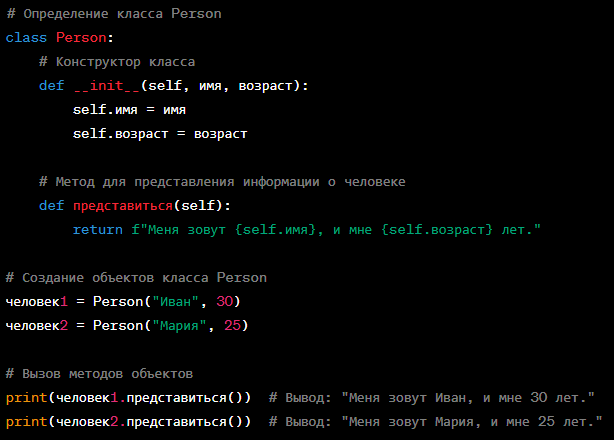


Рисунок 7

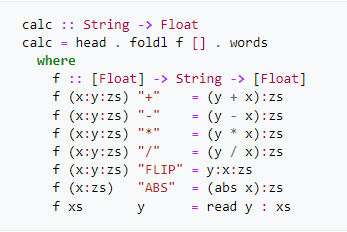


Рисунок 8

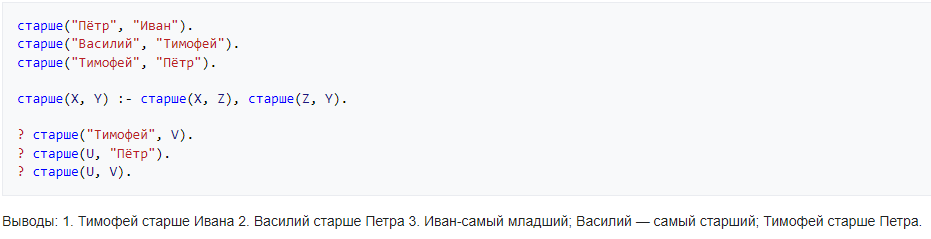


Рисунок 9

3. Сделать выводы о том, какие из парадигм программирования используются наиболее часто.

 4. Для своего индивидуального задания привести обоснование того какие парадигмы  программирования вы будете использовать.

**Задание 3. Изучение правил формирования листинга программ.**

1. Изучить краткие теоретические сведения по теме.(если информация отсутствует в конспекте, то  необходимо ее законспектировать).

2. Оформить листинги программ в соответствии с приведенными рекомендациями по оформлению  программ.

**Необходимо привести примеры оформленного листинга для следующих парадигм  программирования:**

⎯ Структурное программирование (в программе должны использоваться процедуры, функции).

def calculate\_average(numbers):

total = 0

count = 0

for num in numbers:

total += num

count += 1

if count == 0:

return 0 # Защита от деления на ноль

else:

return total / count

# Пример использования функции calculate\_average

my\_numbers = [1, 2, 3, 4, 5]

average = calculate\_average(my\_numbers)

print(f"Среднее значение чисел: {average}")

⎯ Модульное программирование (привести пример оформленного модуля).

# Модуль main.py

# Импортируем функции из модуля rectangle.py

from rectangle import calculate\_area, calculate\_perimeter

# Ввод данных от пользователя

length = float(input("Введите длину прямоугольника: "))

width = float(input("Введите ширину прямоугольника: "))

# Вычисление площади и периметра с использованием функций из модуля rectangle.py

area = calculate\_area(length, width)

perimeter = calculate\_perimeter(length, width)

# Вывод результатов

print(f"Площадь прямоугольника: {area}")

print(f"Периметр прямоугольника: {perimeter}")

⎯ Объектно-ориентированное программирование (привести пример оформленного класса).

class Person:

# Конструктор класса

def init(self, имя, возраст):

self.имя = имя

self.возраст = возраст

# Метод для представления информации о человеке

def представиться(self):

return f"Меня зовут {self.имя}, и мне {self.возраст} лет."

# Создание объектов класса Person

человек1 = Person("Иван", 30)

человек2 = Person("Мария", 25)

# Вызов методов объектов

print(человек1.представиться()) # Вывод: "Меня зовут Иван, и мне 30 лет."

print(человек2.представиться()) # Вывод: "Меня зовут Мария, и мне 25 лет."

**2. Оформить отчет о проделанной работе.**

**3. Защитить работу и сдать ее преподавателю.**

**Контрольные вопросы для защиты лабораторной работы**

**Раздел 6 «Этап реализации»**

1. Дайте определение понятию: **язык программирования. Что такое алфавит,  синтаксис, семантика и стандарт языка программирования?** Приведите примеры известных вам  классификаций языков программирования.
2. Перечислите **критерии выбора языка программирования**. Дайте краткую  характеристику каждому критерию.
3. Дайте определение понятию: **парадигма программирования (стиль программирования)**.  Перечислите известные вам стили программирования.
4. Дайте характеристику следующим **парадигмам программирования**:  (смотри таблицу в задании №2)
5. Что такое **стандарт оформления программного кода.** Обоснуйте необходимость  **качественного формирования листинга программы**. Перечислите правила формирования  листинга программы.
6. Дайте объяснение следующим **правилам оформления кода программы**:
   1. Венгерская нотация
   2. Верблюжья нотация
   3. Грамотный выбор идентификаторов (правила именования переменных,  констант, структур в программе, соглашения о длине имен идентификаторов).
   4. Отступы в программном коде (горизонтальные, вертикальные отступы, отступы  с помощью пробелов)
   5. Использование комментариев
   6. Разбиение задачи на подзадачи
   7. Правила оформления процедур, функций, модулей, классов.

Контрольные вопросы:

1. Язык программирования - это формальная система символов и правил, используемая для написания компьютерных программ. Алфавит - набор символов, синтаксис - правила, определяющие структуру программы, семантика - смысл и действие программы, стандарт - набор соглашений и спецификаций языка. Примеры классификации: высокоуровневые (Python), низкоуровневые (C), функциональные (Haskell), объектно-ориентированные (Java).

2. Критерии выбора языка программирования включают производительность, доступность библиотек, удобство разработки, сообщество разработчиков и применимость к конкретной задаче.

3. Парадигма программирования - это стиль или методология разработки программного кода. Известные стили: императивное, функциональное, объектно-ориентированное.

4. Структурное программирование - организует код в логические блоки с использованием последовательных структур управления (последовательность, ветвление, циклы).

Процедурное программирование - базируется на использовании процедур и функций для организации кода в более мелкие, переиспользуемые части.

Модульное программирование - разделяет программу на модули (независимые компоненты) для упрощения разработки и обслуживания.

Объектно-ориентированное программирование - организует код в объекты, объединяющие данные и методы, что способствует абстракции и инкапсуляции.

Функциональное программирование - основывается на функциях как на основных строительных блоках, поддерживает неизменяемость данных и отсутствие побочных эффектов.

Логическое программирование - использует логические утверждения и инференцию для решения задач, часто ассоциируется с языком Prolog

5. Стандарт оформления программного кода - это набор правил, которые помогают улучшить читаемость и поддерживаемость кода. Необходимость качественного формирования кода включает улучшение совместной разработки и облегчение отладки. Правила формирования включают отступы, выравнивание, комментарии и др.

6. Правила оформления кода:

Венгерская нотация - добавление префиксов к идентификаторам, указывающих тип данных (например, "strName" для строки).

Верблюжья нотация - начало каждого нового слова с большой буквы, без пробелов (например, "camelCase").

Грамотный выбор идентификаторов - понятные имена переменных, констант, структур; соглашения о длине (например, "counter" вместо "c").

Отступы - горизонтальные и вертикальные отступы для лучшей читаемости кода.

Использование комментариев - добавление пояснений к коду для понимания его назначения.

Разбиение задачи на подзадачи - разделение больших задач на более мелкие, что делает код более структурированным.

Правила оформления процедур, функций, модулей, классов - четкое описание и разделение функциональности, использование адекватных имен и документации.