Бойченко Павел ИС-21

Практическая работа № 17_1

Тема:
составление программ с использованием ООП

Цель:
: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с ООП в IDE PyCharm Community.

Постановка задачи:

Создайте класс «Счетчик», который имеет атрибут текущего значения и методы для инкремента и декремента значения

Тип алгоритма:
Линейный

Текст программы:

```
# Создайте класс «Счетчик», который имеет атрибут
текущего значения и методы для
# инкремента и декремента значения


class Counter:
```

```
    def __init__(self, value=0):
        self.value = value

    def increment(self):
        self.value += 1

    def decrement(self):
        self.value -= 1
```

Вывод:
В процессе выполнения практического задания я выработал навыки составления программ линейной структуры в IDE PyCharm Community. Выполнены разработка кода, отладка, тестирование, оптимизация.

Практическая работа № 17_2

Тема:
составление программ с использованием ООП

Цель:
: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с ООП в IDE PyCharm Community.

Постановка задачи:

Создайте класс "Автомобиль", который содержит информацию о марке, модели и годе выпуска. Создайте класс "Грузовик", который наследуется от класса "Автомобиль" и содержит информацию о грузоподъемности. Создайте класс "Легковой автомобиль", который наследуется от класса "Автомобиль" и содержит информацию о количестве пассажиров

Тип алгоритма:
Линейный

Текст программы:

```
# Создайте класс "Автомобиль", который содержит
информацию о марке, модели и
# годе выпуска. Создайте класс "Грузовик", который
наследуется от класса
# "Автомобиль" и содержит информацию о
грузоподъемности. Создайте класс
# "Легковой автомобиль", который наследуется от
класса "Автомобиль" и содержит
# информацию о количестве пассажиров



class Car:
    def __init__(self, brand, model, year):
```

```
        self.brand = brand
        self.model = model
        self.year = year

class Truck(Car):
    def __init__(self, brand, model, year, cargo):
        super().__init__(brand, model, year)
        self.cargo = cargo

class PassengerCar(Car):
    def __init__(self, brand, model, year,
passenger):
        super().__init__(brand, model, year)
        self.passenger = passenger



car = Car("Toyota", "Corolla", 2010)
truck = Truck("Volvo", "FH16", 2020, 5000)
passenger_car = PassengerCar("Honda", "Civic",
2015, 5)
```

Вывод:
В процессе выполнения практического задания я выработал навыки составления программ линейной структуры в IDE PyCharm Community. Выполнены разработка кода, отладка, тестирование, оптимизация.