



ИНТЕЛЛЕКТУАЛЬНЫЙ
МЕГАПОЛИС

ЗАДАЧНИК



Инженерный класс

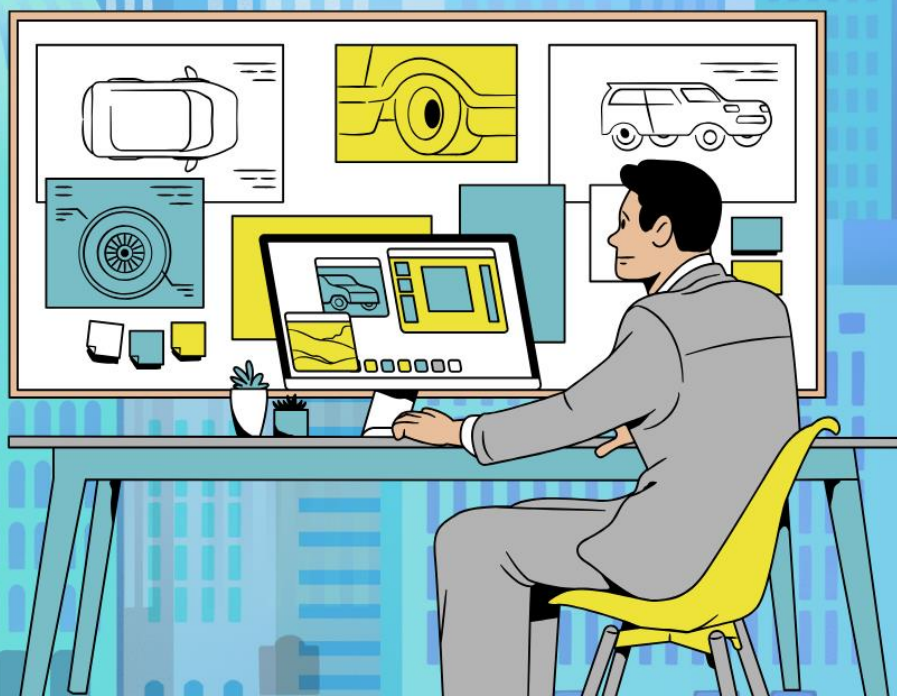
В МОСКОВСКОЙ ШКОЛЕ

НАПРАВЛЕНИЕ
АВИАСТРОИТЕЛЬНЫЕ КЛАССЫ

ПРАКТИЧЕСКИЙ ЭТАП



МОСКВА
2024



СОДЕРЖАНИЕ

Вариант 1	3
Вариант 2	10
Вариант 3	16
Вариант 4	23
Вариант 5	29
Вариант 6	36
Вариант 7	43
Вариант 8	50

Вариант 1
Кейс №1 «Программирование».

«Программа управления библиотекой»

Разработать на языке Python программу с консольным интерфейсом для системы управления библиотекой. Программа должна поддерживать работу с книгами, позволяя добавлять книги, удалять их из каталога, просматривать список всех книг, а также искать книги по названию. Каждая книга имеет уникальный идентификационный номер, название и количество экземпляров.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_book	название: str, идентификационный номер: int, количество: int	Добавляет книгу в каталог. Если книга с таким идентификационным номером уже существует, увеличивает её количество.	add_book("Гарри Поттер", 101, 3)
remove_book	идентификационный номер: int	Удаляет книгу из каталога по её идентификационному номеру. Если книги нет, выводит сообщение об ошибке.	remove_book(101)
view_catalog	Нет параметров	Выводит список всех книг в каталоге с указанием количества экземпляров каждой книги.	view_catalog()
search_book	название книги: str	Ищет книгу по названию и выводит её идентификационный номер и количество экземпляров.	search_book("Гарри Поттер")

Хранение данных:

- Все данные о книгах хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название книги и количество экземпляров.

- Идентификационный номер и количество являются положительными целыми числами.
- Название - данное строчное значение является уникальным для каждой книги в библиотеке.
- Идентификационный номер - данное целое числовое значение является уникальным для каждой книги в библиотеке. Это означает, что для каждой книги есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных о книгах

```
books = {
    101: ("Гарри Поттер", 3),
    102: ("Война и мир", 2)
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():
    while True:
        print("Добро пожаловать в библиотечную систему!")
        print("1. Добавить книгу в каталог")
        print("2. Удалить книгу из каталога")
        print("3. Просмотреть каталог книг")
        print("4. Найти книгу")
        print("5. Выйти")
        choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление книги

Ввод:

Выберите действие: 1

Введите название книги: Гарри Поттер

Введите идентификационный номер книги: 101

Введите количество экземпляров книги: 3

Ожидаемый результат:

Книга 'Гарри Поттер' добавлена успешно.

Дополнительный ввод (если книга уже существует):

Выберите действие: 1

Введите название книги: Гарри Поттер

Введите идентификационный номер книги: 101

Введите количество экземпляров книги: 3

Ожидаемый результат:

Информация о книге 'Гарри Поттер' обновлена. Количество экземпляров в каталоге: 6.

Пример 2: Удаление книги

Ввод:

Выберите действие: 2

Введите идентификационный номер книги: 101

Ожидаемый результат:

Книга удалена успешно.

Если книги нет в системе:

Выберите действие: 2

Введите идентификационный номер книги: 9999

Ожидаемый результат:

Ошибка: Книга с идентификационным номером 9999 не найдена.

Пример 3: Просмотр всех книг

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список книг в библиотеке:

Идентификационный номер: 101

Название: Гарри Поттер

Количество: 6

Пример 4: Поиск книги

Ввод:

Выберите действие: 4

Введите название книги: Гарри Поттер

Ожидаемый результат:

Идентификационный номер книги: 101

Количество в библиотеке: 6

Если книга не найден:

Выберите действие: 4

Введите название книги: Пикник на обочине

Ожидаемый результат:

Ошибка: Книга 'Пикник на обочине' не найдена.

Кейс №2 «3D-моделирование и 3D-печать».

Задание 1

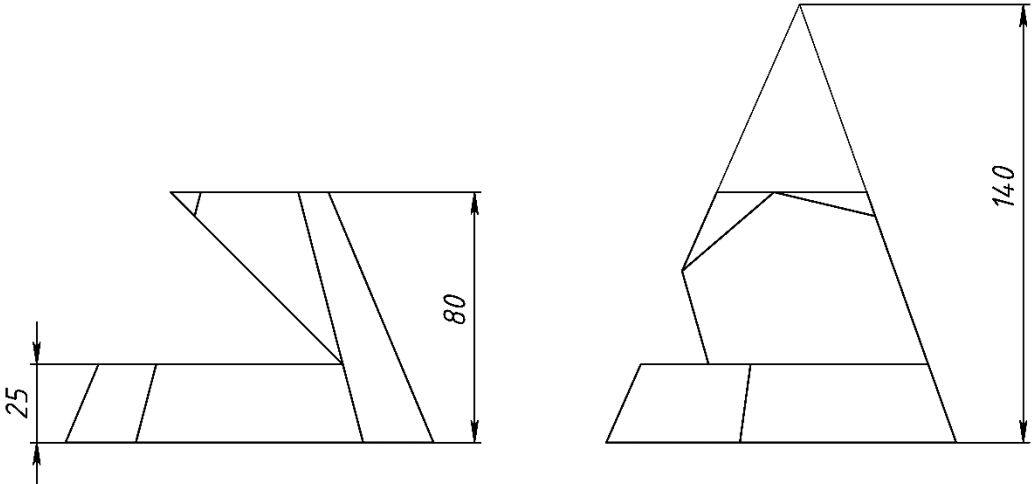
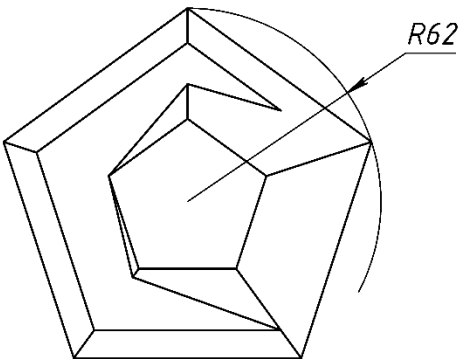
Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Перв. примен.		Справ. №		Вариант 1				
								
Подп. и дата		Инв. № дубл.						
Взам. инв. №		Инв. № подл.		Вариант 1				
Подп. и дата				Примитив				
Изм.		Лист	№ докум.	Подп.	Дата	Лит.	Масса	Масштаб
Разраб.								1:2
Пров.						Лист	Листов 1	
Т. контр.						МАИ НИУ		
Н. контр.								
Утв.								

Копировал _____ Формат А4

Рисунок 1

Справ. №		Перв. примен.		Вариант 1																		
				Отверстие сквозное																		
				Вариант 1																		
				Деталь					<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">Лит.</td> <td style="width: 10%;">Масса</td> <td style="width: 10%;">Масштаб</td> </tr> <tr> <td style="height: 20px;"></td> <td style="height: 20px;"></td> <td style="text-align: center;">1:5</td> </tr> <tr> <td colspan="2" style="text-align: center;">Лист</td> <td style="text-align: center;">Листов 1</td> </tr> </table>					Лит.	Масса	Масштаб			1:5	Лист		Листов 1
				Лит.	Масса	Масштаб																
		1:5																				
Лист		Листов 1																				
МАИ НИУ																						
Инв. № подл.	Подп. и дата		Взам. инв. №	Инв. № дубл.	Подп. и дата																	
Изм.	Лист	№ докум.	Подп.	Дата																		
Разраб.																						
Пров.																						
Т. контр.																						
Н. контр.																						
Утв.																						

Копировал

Формат А4

Рисунок 2

Вариант 2

Кейс №1 «Программирование».

«Программа управления автопарком»

Разработать на языке Python программу с консольным интерфейсом для системы управления автопарком компании. Программа должна поддерживать учет транспортных средств, позволяя добавлять машины, удалять их из автопарка, просматривать список всех автомобилей и находить их по модели.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_vehicle	марка: str, идентификационный номер: int, количество: int	Добавляет автомобиль в автопарк. Если автомобиль с таким номером уже существует, увеличивает его количество.	add_vehicle("Toyota", 301, 5)
remove_vehicle	идентификационный номер: int	Удаляет автомобиль из автопарка по его идентификационному номеру. Если автомобиля нет, выводит сообщение об ошибке.	remove_vehicle(301)
view_fleet	Нет параметров	Выводит список всех автомобилей в автопарке с указанием количества каждой модели.	view_fleet()
search_vehicle	марка: str	Ищет автомобиль по марке и выводит его идентификационный номер и количество машин в автопарке.	search_vehicle("Toyota")

Хранение данных:

- Все данные о машинах хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий марку авто и количество автомобилей данной марки.
- Идентификационный номер и количество являются положительными целыми числами.

- Название - данное строчное значение является уникальным для каждого автомобиля в автопарке.
- Идентификационный номер - данное целое числовое значение является уникальным для каждого автомобиля в автопарке. Это означает, что для каждого автомобиля есть уникальная пара значений (Название марки - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных об автомобилях

```
vehicles = {
    301: ("Toyota", 5),
    302: ("Ford", 3)
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():
    while True:
        print("Добро пожаловать в систему управления автопарком!")
        print("1. Добавить автомобиль в автопарк")
        print("2. Удалить автомобиль из автопарка")
        print("3. Просмотреть автопарк")
        print("4. Найти автомобиль")
        print("5. Выйти")
        choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление автомобиля

Ввод:

Выберите действие: 1

Введите марку автомобиля: Toyota

Введите идентификационный номер: 301

Введите количество: 5

Ожидаемый результат:

Автомобиль 'Toyota' добавлен успешно.

Дополнительный ввод (если марка уже существует):

Выберите действие: 1

Введите марку автомобиля: Toyota

Введите идентификационный номер: 301

Введите количество: 5

Ожидаемый результат:

Информация о марке 'Toyota' обновлена. Количество машин в автопарке: 10.

Пример 2: Удаление автомобиля

Ввод:

Выберите действие: 2

Введите идентификационный номер автомобиля: 301

Ожидаемый результат:

Автомобиль удален успешно.

Если марки автомобиля нет в системе:

Выберите действие: 2

Введите идентификационный номер автомобиля: 9999

Ожидаемый результат:

Ошибка: Автомобиль с идентификационным номером 9999 не найден.

Пример 3: Просмотр всего автопарка

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список автомобилей в автопарке:

Идентификационный номер: 301

Название: Toyota

Количество: 12

Пример 4: Поиск автомобиля

Ввод:

Выберите действие: 4

Введите название марки автомобиля: Toyota

Ожидаемый результат:

Идентификационный номер марки автомобиля: 301

Количество в автопарке: 12

Если марка автомобиля не найдена:

Выберите действие: 4

Введите название марки: Lada

Ожидаемый результат:

Ошибка: Автомобили марки 'Lada' не найдены

Кейс №2 «3D-моделирование и 3D-печать».

Задание 1

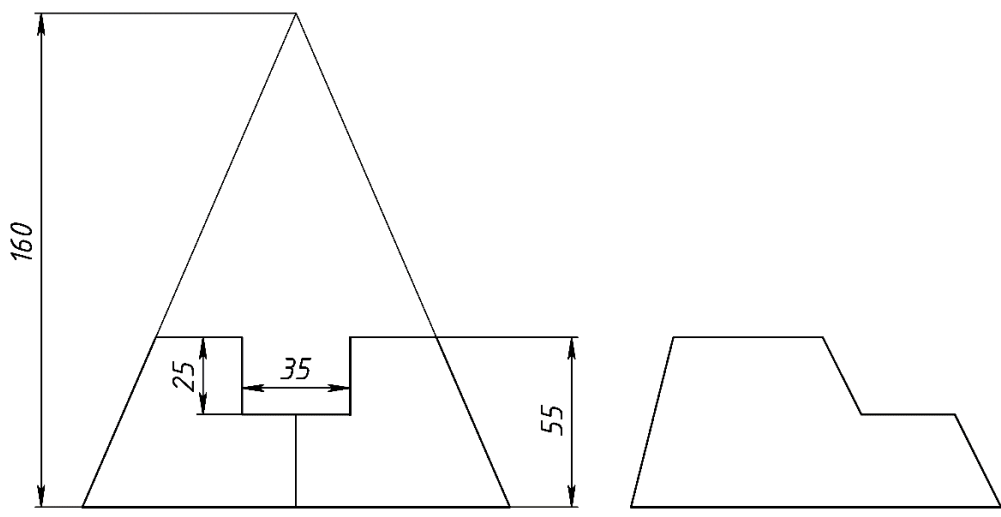
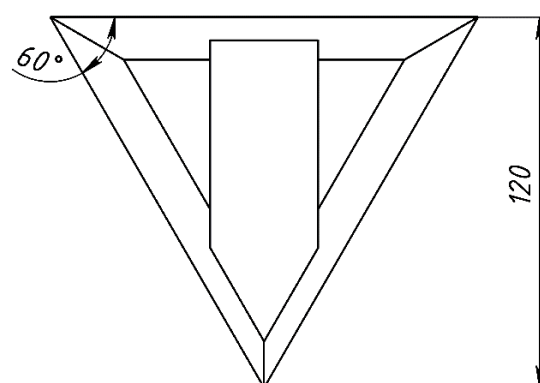
Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Справ. №		Перв. примен.		Вариант 2					
									
									
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант 2				
	Изм.	Лист	№ докум.	Подп.	Дата	Примитив	Лит.	Масса	Масштаб
	Разраб.								1:5
	Пров.						Лист	Листов 1	
	Т. контр.						МАИ НИУ		
Н. контр.									
Утв.									

Копировал

Формат А4

Рисунок 1

Инф. № подл.		Подп. и дата		Взам. инф. №		Инф. № докл.		Подп. и дата		Справ. №		Перв. примен.	
												Вариант 2	

Вариант 2								
Деталь	Изм.	Лист	№ докум.	Подп.	Дата	Лит.	Масса	Масштаб
	Разраб.							1:2
	Пров.							
	Т. контр.					Лист	Листов 1	
	Н. контр.					МАИ НИУ		
Утв.								

Копировал

Формат А4

Рисунок 2

Вариант 3
Кейс №1 «Программирование».

«Программа управления рестораном»

Разработать на языке Python программу с консольным интерфейсом для системы управления кухней ресторана. Программа должна поддерживать учет блюд, позволяя добавлять новые блюда в меню, удалять блюда, просматривать текущее меню, а также искать блюда по названию.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_dish	название: str, идентификационный номер: int, количество порций: int	Добавляет новое блюдо в меню. Если блюдо с таким номером уже существует, увеличивает количество порций.	add_dish("Пицца", 401, 20)
remove_dish	идентификационный номер: int	Удаляет блюдо из меню по его идентификационному номеру. Если блюда нет, выводит сообщение об ошибке.	remove_dish(401)
view_menu	Нет параметров	Выводит список всех блюд в меню с указанием количества порций для каждого блюда.	view_menu()
search_dish	название блюда: str	Ищет блюдо по названию и выводит его идентификационный номер и количество порций.	search_dish("Пицца")

Хранение данных:

- Все данные о позициях блюд хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название блюда и количество порций.
- Идентификационный номер и количество являются положительными целыми числами.
- Название - данное строчное значение является уникальным для каждого блюда в ресторане.
- Идентификационный номер - данное целое числовое значение является уникальным для каждого блюда в ресторане. Это означает, что для каждого блюда есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных о блюдах

```
dishes = {  
    401: ("Пицца", 20),  
    402: ("Салат", 15)  
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():  
  
    while True:  
  
        print("Добро пожаловать в систему управления рестораном!")  
  
        print("1. Добавить блюдо в меню")  
  
        print("2. Удалить блюдо из меню")  
  
        print("3. Просмотреть меню")  
  
        print("4. Найти блюдо")  
  
        print("5. Выйти")  
  
        choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление блюда

Ввод:

Выберите действие: 1

Введите название блюда: Пицца

Введите идентификационный номер: 401

Введите количество порций: 20

Ожидаемый результат:

Блюдо 'Пицца' добавлено успешно.

Дополнительный ввод (если блюдо уже существует):

Выберите действие: 1

Введите название товара: Пицца

Введите идентификационный номер блюда: 401

Введите количество порций: 5

Ожидаемый результат:

Блюдо 'Пицца' обновлено. Количество порций: 25.

Пример 2: Удаление блюда

Ввод:

Выберите действие: 2

Введите идентификационный номер блюда: 401

Ожидаемый результат:

Блюдо удалено успешно.

Если блюда нет в системе:

Выберите действие: 2

Введите идентификационный номер блюда: 9999

Ожидаемый результат:

Ошибка: Блюдо с идентификационным номером 9999 не найдено.

Пример 3: Просмотр всех блюд и порций

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список блюд:

Идентификационный номер: 401

Название: Пицца

Количество: 12

Пример 4: Поиск блюда**Ввод:**

Выберите действие: 4

Введите название блюда: Пицца

Ожидаемый результат:

Идентификационный номер блюда: 401

Количество порций: 12

Если блюдо не найдено:

Выберите действие: 4

Введите название блюда: Сэндвич

Ожидаемый результат:

Ошибка: Блюдо 'Сэндвич' не найдено.

Кейс №2 «3D-моделирование и 3D-печать».***Задание 1***

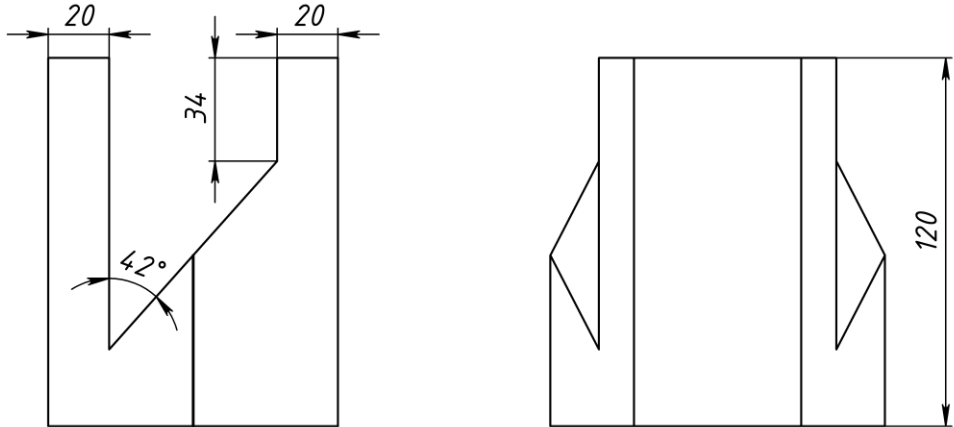
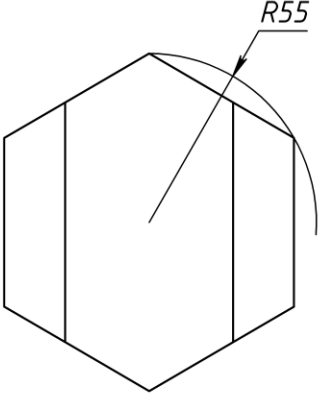
Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Перв. примен.		Справ. №		Вариант 3																																		
																																						
																																						
Подп. и дата		Инв. № дубл.		Взам. инв. №		Вариант 3																																
						<table border="1"> <tr> <td>Изм.</td> <td>Лист</td> <td>№ докум.</td> <td>Подп.</td> <td>Дата</td> </tr> <tr> <td>Разраб.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Пров.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Т. контр.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Н. контр.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Утв.</td> <td></td> <td></td> <td></td> <td></td> </tr> </table>			Изм.	Лист	№ докум.	Подп.	Дата	Разраб.					Пров.					Т. контр.					Н. контр.					Утв.				
Изм.	Лист	№ докум.	Подп.	Дата																																		
Разраб.																																						
Пров.																																						
Т. контр.																																						
Н. контр.																																						
Утв.																																						
Инв. № подл.						<table border="1"> <tr> <td colspan="3">Лит.</td> <td>Масса</td> <td>Масштаб</td> </tr> <tr> <td colspan="3"></td> <td></td> <td>1:2</td> </tr> <tr> <td colspan="2">Лист</td> <td colspan="3">Листов 1</td> </tr> <tr> <td colspan="5">МАИ НИУ</td> </tr> </table>			Лит.			Масса	Масштаб					1:2	Лист		Листов 1			МАИ НИУ														
Лит.			Масса	Масштаб																																		
				1:2																																		
Лист		Листов 1																																				
МАИ НИУ																																						

Копировал

Формат А4

Рисунок 1

Перв. примен.	
Справ. №	

Вариант 3

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.				
Пров.				
Т. контр.				
Н. контр.				
Утв.				

Вариант 3

Деталь

Лит.	Масса	Масштаб
		1:2
Лист		Листов 1
МАИ НИУ		

Копировал

Формат А4

Рисунок 2

Вариант 4
Кейс №1 «Программирование».

«Программа управления интернет-магазина видеоигр»

Разработать на языке Python программу с консольным интерфейсом для системы управления интернет-магазина видеоигр. Программа должна позволять добавлять игры в магазин, удалять игры, просматривать список доступных игр, а также искать игры по названию.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_game	название: str, идентификационный номер: int, количество копий: int	Добавляет новую игру в каталог. Если игра с таким номером уже существует, увеличивает количество копий.	add_game("Minecraft", 501, 10)
remove_game	идентификационный номер: int	Удаляет игру из каталога по её идентификационному номеру. Если игры нет, выводит сообщение об ошибке.	remove_game(501)
view_catalog	Нет параметров	Выводит список всех игр в каталоге с указанием количества копий каждой игры.	view_catalog()
search_game	название игры: str	Ищет игру по названию и выводит её идентификационный номер и количество копий.	search_game("Minecraft")

Хранение данных:

- Все данные о видеоиграх хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название игры и количество ее экземпляров.
- Идентификационный номер и количество являются положительными целыми числами.

- Название - данное строчное значение является уникальным для каждой игры в магазине.
- Идентификационный номер - данное целое числовое значение является уникальным для каждой игры в магазине. Это означает, что для каждой игры есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных об играх

```
games = {
    501: ("Minecraft", 10),
    502: ("The Witcher 3", 7)
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():
    while True:
        print("Добро пожаловать в интернет-магазин видеоигр!")
        print("1. Добавить игру в каталог")
        print("2. Удалить игру из каталога")
        print("3. Просмотреть каталог")
        print("4. Найти игру")
        print("5. Выйти")
        choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление игры

Ввод:

Выберите действие: 1

Введите название игры: Minecraft

Введите идентификационный номер: 501

Введите количество копий: 10

Ожидаемый результат:

Игра 'Minecraft' добавлена успешно.

Дополнительный ввод (если игра уже существует):

Выберите действие: 1

Введите название игры: Minecraft

Введите идентификационный номер: 501

Введите количество копий: 10

Ожидаемый результат:

Информация об игре 'Minecraft' обновлена. Количество на складе: 20.

Пример 2: Удаление игры

Ввод:

Выберите действие: 2

Введите идентификационный номер игры: 501

Ожидаемый результат:

Игра удалена успешно.

Если игры нет в системе:

Выберите действие: 2

Введите идентификационный номер игры: 9999

Ожидаемый результат:

Ошибка: Игра с идентификационным номером 9999 не найдена.

Пример 3: Просмотр каталога

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список игр в магазине:

Идентификационный номер: 501

Название: Minecraft

Количество: 20

Пример 4: Поиск игры**Ввод:**

Выберите действие: 4

Введите название игры: Minecraft

Ожидаемый результат:

Идентификационный номер товара: 501

Количество в магазине: 20

Если товар не найден:

Выберите действие: 4

Введите название игры: Hotline Miami

Ожидаемый результат:

Ошибка: Игра 'Hotline Miami' не найдена.

Кейс №2 «3D-моделирование и 3D-печать».**Задание 1**

Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Инв. № подл.		Подп. и дата		Взам. инв. №		Инв. № дубл.		Подп. и дата		Справ. №		Перв. примен.	
												Вариант 4	

Вариант 4				
Изм.	Лист	№ докум.	Подп.	Дата
Разраб.				
Пров.				
Т. контр.				
Н. контр.				
Утв.				

Деталь			Лит.	Масса	Масштаб
1:2					
Лист			Листов 1		
МАИ НИУ					

Копировал
Формат А4

Рисунок 2

Вариант 5
Кейс №1 «Программирование».

«Программа управления зоопарком»

Разработать на языке Python программу с консольным интерфейсом для системы управления зоопарком. Программа должна позволять добавлять животных в зоопарк, удалять их, просматривать список всех животных, а также искать животных по названию.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_animal	название: str, идентификационный номер: int, количество экземпляров: int	Добавляет животное в зоопарк. Если животное с таким номером уже существует, увеличивает количество экземпляров.	add_animal("Лев", 2001, 3)
remove_animal	идентификационный номер: int	Удаляет животное из зоопарка по его идентификационному номеру. Если животное не найдено, выводит сообщение об ошибке.	remove_animal(2001)
view_zoo	Нет параметров	Выводит список всех животных в зоопарке с указанием их количества.	view_zoo()
search_animal	название животного: str	Ищет животное по названию и выводит его идентификационный номер и количество экземпляров.	search_animal("Лев")

Хранение данных:

- Все данные о животных хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название животного и его количество.
- Идентификационный номер и количество являются положительными целыми числами.
- Название - данное строчное значение является уникальным для каждого животного в зоопарке.
- Идентификационный номер - данное целое числовое значение является уникальным для каждого животного в зоопарке. Это означает, что для каждого животного есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных о животных

```
animals = {  
    2001: ("Лев", 3),  
    2002: ("Тигр", 5)  
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():  
  
    def main():  
  
        while True:  
  
            print("Добро пожаловать в систему управления зоопарком!")  
  
            print("1. Добавить животное")  
  
            print("2. Удалить животное")  
  
            print("3. Просмотреть список животных")  
  
            print("4. Найти животное")  
  
            print("5. Выйти")  
  
            choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление животного

Ввод:

Выберите действие: 1

Введите название животного: Лев

Введите идентификационный номер животного: 2001

Введите количество экземпляров животного: 3

Ожидаемый результат:

Животное 'Лев' добавлено успешно.

Если животное уже существует:

Выберите действие: 1

Введите название животного: Лев

Введите идентификационный номер животного: 2001

Введите количество экземпляров животного: 2

Ожидаемый результат:

Информация о животном 'Лев' обновлена. Количество экземпляров: 5.

Пример 2: Удаление животного

Ввод:

Выберите действие: 2

Введите идентификационный номер животного: 2001

Ожидаемый результат:

Животное удалено успешно.

Если животное не найдено:

Выберите действие: 2

Введите идентификационный номер животного: 9999

Ожидаемый результат:

Ошибка: Животное с идентификационным номером 9999 не найдено.

Пример 3: Просмотр всех животных

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список животных в зоопарке:

Идентификационный номер: 2001

Название: Лев

Количество: 5

Идентификационный номер: 2002

Название: Тигр

Количество: 5

Пример 4: Поиск животного**Ввод:**

Выберите действие: 4

Введите название животного: Лев

Ожидаемый результат:

Идентификационный номер животного: 2001

Количество экземпляров в зоопарке: 5

Если животное не найдено:

Выберите действие: 4

Введите название животного: Жираф

Ожидаемый результат:

Ошибка: Животное 'Жираф' не найдено.

Кейс №2 «3D-моделирование и 3D-печать».**Задание 1**

Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР).

Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Вариант 6
Кейс №1 «Программирование».

«Программа управления садом растений»

Разработать на языке Python программу с консольным интерфейсом для системы управления садом растений. Программа должна позволять добавлять растения в сад, удалять их, просматривать список всех растений, а также искать их по названию.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_plant	название: str, идентификационный номер: int, количество экземпляров: int	Добавляет новое растение в сад. Если растение с таким номером уже существует, увеличивает количество экземпляров.	add_plant("Роза", 801, 12)
remove_plant	идентификационный номер: int	Удаляет растение из сада по его идентификационному номеру. Если растение не найдено, выводит сообщение об ошибке.	remove_plant(801)
view_garden	Нет параметров	Выводит список всех растений в саду с указанием их количества.	view_garden()
search_plant	название растения: str	Ищет растение по названию и выводит его идентификационный номер и количество экземпляров.	search_plant("Роза")

Хранение данных:

- Все данные о растениях хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название растения и количество экземпляров.
- Идентификационный номер и количество являются положительными целыми числами.
- Название - данное строчное значение является уникальным для каждого название растения.
- Идентификационный номер - данное целое числовое значение является уникальным для каждого название растения. Это означает, что для каждого название растения есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных о растениях

```
plants = {  
    801: ("Роза", 12),  
    802: ("Тюльпан", 20)  
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():  
  
    def main():  
  
        while True:  
  
            print("Добро пожаловать в систему управления садом!")  
  
            print("1. Добавить растение")  
  
            print("2. Удалить растение")  
  
            print("3. Просмотреть список растений")  
  
            print("4. Найти растение")  
  
            print("5. Выйти")  
  
            choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление растения

Ввод:

Выберите действие: 1

Введите название растения: Роза

Введите идентификационный номер: 801

Введите количество экземпляров: 12

Ожидаемый результат:

Растение 'Роза' добавлено успешно.

Дополнительный ввод (если растение уже существует):

Выберите действие: 1

Введите название растения: Роза

Введите идентификационный номер: 801

Введите количество экземпляров: 3

Ожидаемый результат:

Информация о растении 'Роза' обновлена. Количество на складе: 15.

Пример 2: Удаление растения

Ввод:

Выберите действие: 2

Введите идентификационный номер растения: 801

Ожидаемый результат:

Растение удалено успешно.

Если растения нет в системе:

Выберите действие: 2

Введите идентификационный номер растения: 9999

Ожидаемый результат:

Ошибка: Растение с идентификационным номером 9999 не найден.

Пример 3: Просмотр всех растений

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список растений в саду:

Идентификационный номер: 801

Название: Роза

Количество: 12

Пример 4: Поиск растения

Ввод:

Выберите действие: 4

Введите название растения: Роза

Ожидаемый результат:

Идентификационный номер растения: 801

Количество в саду: 12

Если растение не найден:

Выберите действие: 4

Введите название растения: Герань

Ожидаемый результат:

Ошибка: Растение 'Герань' не найдено.

Кейс №2 «3D-моделирование и 3D-печать».

Задание 1

Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху, Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

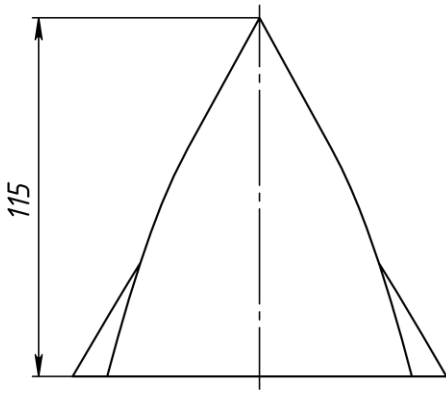
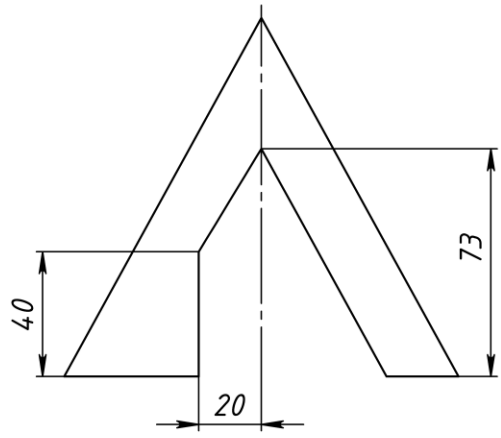
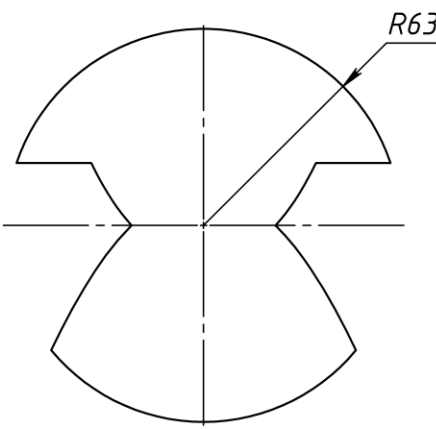
Перв. примен.		Справ. №		Вариант 6					
									
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант 6				
	Изм.	Лист	№ докум.	Подп.					
	Примитив					Лит.	Масса	Масштаб	
								1:2	
						Лист	Листов 1		
						МАИ НИУ			
	Разраб.					Копировал			Формат А4
Пров.									
Т. контр.									
Н. контр.									
Утв.									

Рисунок 1

Инв. № подл.		Подп. и дата		Взам. инв. №		Инв. № дубл.		Подп. и дата		Справ. №		Перв. примен.																													
Вариант 6																																									
<table border="1"> <tr> <td>Изм.</td> <td>Лист</td> <td>№ докум.</td> <td>Подп.</td> <td>Дата</td> </tr> <tr> <td>Разраб.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Пров.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Т. контр.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Н. контр.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Утв.</td> <td></td> <td></td> <td></td> <td></td> </tr> </table>						Изм.	Лист	№ докум.	Подп.	Дата	Разраб.					Пров.					Т. контр.					Н. контр.					Утв.					Вариант 6 Деталь					
Изм.	Лист	№ докум.	Подп.	Дата																																					
Разраб.																																									
Пров.																																									
Т. контр.																																									
Н. контр.																																									
Утв.																																									
						Лист		Масса		Масштаб																															
										1:2																															
						Лист		Листов 1																																	
						МАИ НИУ																																			

Копировал
Формат А4

Рисунок 2

Вариант 7
Кейс №1 «Программирование».

«Программа управления музеем»

Разработать на языке Python программу с консольным интерфейсом для системы управления экспонатами музея. Программа должна позволять добавлять экспонаты в коллекцию музея, удалять их, просматривать список всех экспонатов, а также искать их по названию.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_exhibit	название: str, идентификационный номер: int, количество экземпляров: int	Добавляет новый экспонат в коллекцию музея. Если экспонат с таким номером уже существует, увеличивает количество.	add_exhibit("Статуя", 901, 3)
remove_exhibit	идентификационный номер: int	Удаляет экспонат из коллекции по его идентификационному номеру. Если экспонат не найден, выводит сообщение об ошибке.	remove_exhibit(901)
view_museum	Нет параметров	Выводит список всех экспонатов в коллекции с указанием их количества.	view_museum()
search_exhibit	название экспоната: str	Ищет экспонат по названию и выводит его идентификационный номер и количество экземпляров.	search_exhibit("Статуя")

Хранение данных:

- Все данные об экспонатах хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер, а значением - кортеж, содержащий название экспоната и количество экземпляров.
- Идентификационный номер и количество являются положительными целыми числами.
- Название - данное строчное значение является уникальным для каждого названия экспоната.
- Идентификационный номер - данное целое числовое значение является уникальным для каждого экспоната. Это означает, что для каждого экспоната есть уникальная пара значений (Название - Идентификационный номер), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных об экспонатах

```
exhibits = {  
    901: ("Статуя", 3),  
    902: ("Картина", 5)  
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():  
  
    def main():  
  
        while True:  
  
            print("Добро пожаловать в систему управления экспонатами музея!")  
  
            print("1. Добавить экспонат")  
  
            print("2. Удалить экспонат")  
  
            print("3. Просмотреть список экспонатов")  
  
            print("4. Найти экспонат")  
  
            print("5. Выйти")  
  
            choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление экспоната

Ввод:

Выберите действие: 1

Введите название экспоната: Статуя

Введите идентификационный номер: 901

Введите количество экземпляров: 3

Ожидаемый результат:

Экспонат 'Статуя' добавлен успешно.

Дополнительный ввод (если экспонат уже существует):

Выберите действие: 1

Введите название экспоната: Статуя

Введите идентификационный номер: 901

Введите количество экземпляров: 12

Ожидаемый результат:

Информация об экспонате 'Статуя' обновлена. Количество в музее: 15.

Пример 2: Удаление экспоната

Ввод:

Выберите действие: 2

Введите идентификационный номер экспоната: 901

Ожидаемый результат:

Экспонат удален успешно.

Если экспоната нет в системе:

Выберите действие: 2

Введите идентификационный номер экспоната: 9999

Ожидаемый результат:

Ошибка: Экспонат с идентификационным номером 9999 не найден.

Пример 3: Просмотр всех экспонатов

Ввод:

Выберите действие: 3

Ожидаемый результат:

Список экспонатов:

Идентификационный номер: 901

Название: Статуя

Количество: 3

Пример 4: Поиск экспоната

Ввод:

Выберите действие: 4

Введите название экспоната: Статуя

Ожидаемый результат:

Идентификационный номер экспоната: 901

Количество: 3

Если экспонат не найден:

Выберите действие: 4

Введите название экспоната: Чаша

Ожидаемый результат:

Ошибка: Экспонат 'Чаша' не найден.

Кейс №2 «3D-моделирование и 3D-печать».

Задание 1

Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Перв. примен.		Справ. №		Вариант 7	
Подп. и дата		Взам. инв. №		Инв. № дудл.	
Подп. и дата		Взам. инв. №		Инв. № дудл.	
Инв. № подл.		Изм.		Лист	
Т. контр.		Пров.		Разраб.	
Н. контр.		Утв.		№ докум.	
				Подп.	
				Дата	
				Вариант 7 Примитив	
Лит.		Масса		Масштаб	
Лист		Листов 1		1:2	
				МАИ НИУ	

Копировал

Формат А4

Рисунок 1

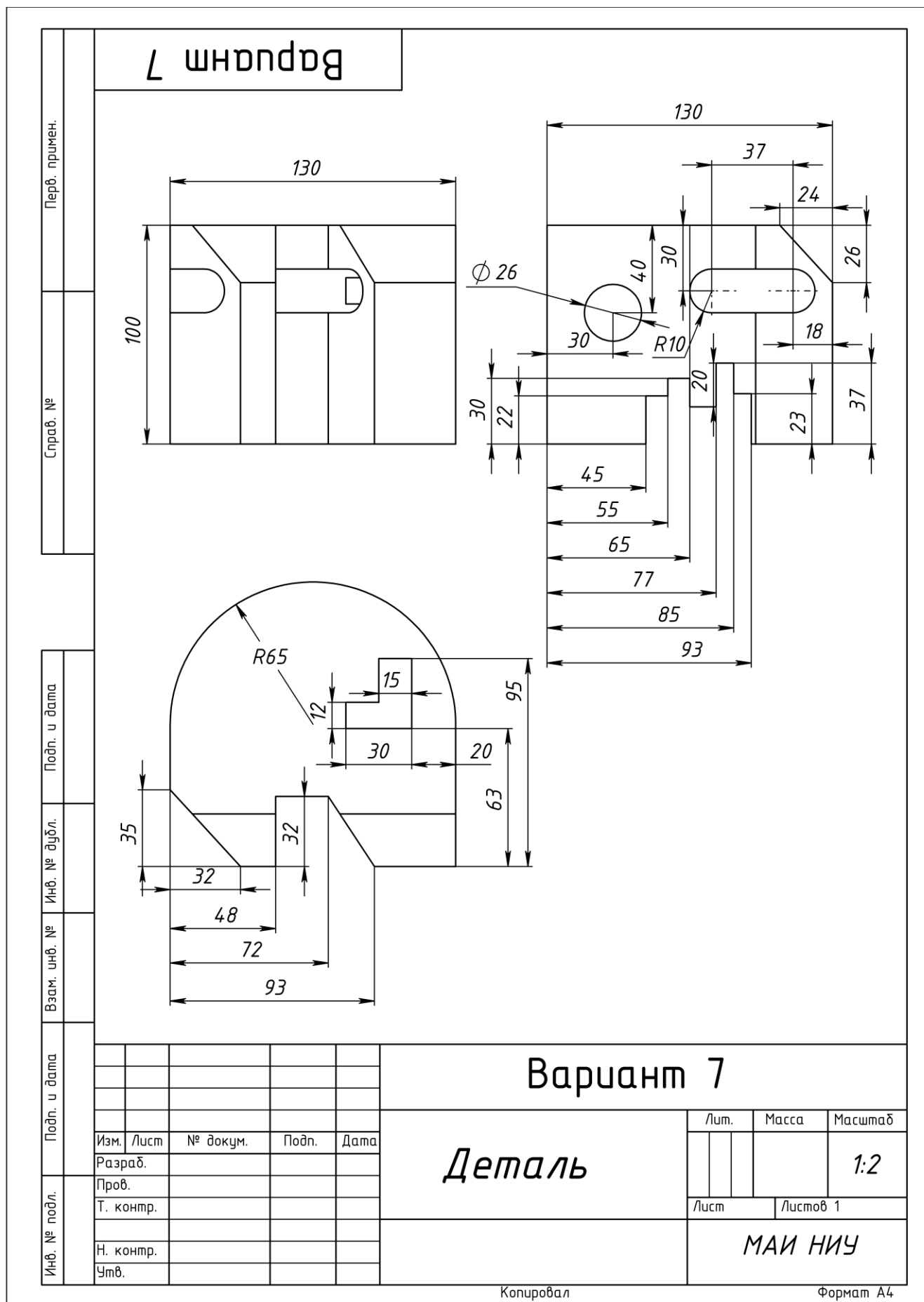


Рисунок 2

Вариант 8
Кейс №1 «Программирование».

«Программа управления парковкой»

Разработать на языке Python программу с консольным интерфейсом для системы управления парковочными местами. Программа должна позволять добавлять автомобили на парковку, удалять их, просматривать список занятых мест, а также искать автомобили по номерному знаку.

При написании программы каждая команда реализуется в виде отдельной функции, все функции пишутся в одном общем файле вместе с основной программой.

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
add_vehicle	номерной знак: str, идентификационный номер парковочного места: int	Добавляет автомобиль на парковку. Если автомобиль с таким номером уже есть, выводит сообщение об ошибке.	add_vehicle("A123BK77", 1001)
remove_vehicle	идентификационный номер парковочного места: int	Удаляет автомобиль с парковки по его идентификационному номеру парковочного места. Если автомобиля нет, выводит сообщение об ошибке.	remove_vehicle(1001)
view_parking	Нет параметров	Выводит список всех занятых парковочных мест с указанием номерных знаков автомобилей.	view_parking()
search_vehicle	номерной знак: str	Ищет автомобиль по номерному знаку и выводит его идентификационный номер парковочного места.	search_vehicle("A123BK77")

Хранение данных:

- Все данные об автомобилях хранятся в виде глобальной переменной в виде словаря, где ключом является уникальный идентификационный номер парковочного места, а значением - номерной знак.
- Идентификационный номер парковочного места является положительным целым числом .
- Номерной знак - данное строчное значение является уникальным для каждого автомобиля.
- Идентификационный номер парковочного места - данное целое числовое значение является уникальным для каждого места. Это означает, что для каждого автомобиля есть уникальная пара значений (Номерной знак - Идентификационный номер парковки), дополнительные проверки реализовать не нужно.
- Указание типов входных параметров в явном виде необязательно.
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.

Пример хранения данных об автомобилях на парковке

```
parking = {  
    1001: "A123BK77",  
    1002: "B456PT99"  
}
```

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем.

Примерная структура интерфейса:

```
def main():  
  
    def main():  
  
        while True:  
  
            print("Добро пожаловать в систему управления парковкой!")  
  
            print("1. Добавить автомобиль на парковку")  
  
            print("2. Удалить автомобиль с парковки")  
  
            print("3. Просмотреть список занятых мест")  
  
            print("4. Найти автомобиль по номерному знаку")  
  
            print("5. Выйти")  
  
            choice = int(input("Выберите действие: "))
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы

Пример 1: Добавление автомобиля

Ввод:

Выберите действие: 1

Введите номерной знак автомобиля: A123BK77

Введите идентификационный номер места: 1001

Ожидаемый результат:

Автомобиль с номером 'A123BK77' добавлен на парковку.

Если место уже занято другим автомобилем:

Выберите действие: 1

Введите номерной знак автомобиля: B456CD78

Введите идентификационный номер места: 1001

Ожидаемый результат:

Ошибка: Место с идентификационным номером 1001 уже занято.

Пример 2: Удаление автомобиля

Ввод:

Выберите действие: 2

Введите идентификационный номер места: 1001

Ожидаемый результат:

Автомобиль удален с парковки.

Если место пустое:

Выберите действие: 2

Введите идентификационный номер места: 1002

Ожидаемый результат:

Ошибка: Место с идентификационным номером 1002 пустое.

Пример 3: Просмотр всех занятых мест

Ввод:

Выберите действие: 3

Ожидаемый результат:

Занятые парковочные места:

Идентификационный номер места: 1001

Номерной знак: A123BK77

Пример 4: Поиск автомобиля

Ввод:

Выберите действие: 4

Введите номерной знак автомобиля: A123BK77

Ожидаемый результат:

Идентификационный номер места: 1001

Если автомобиль не найден:

Выберите действие: 4

Введите номерной знак автомобиля: C789DE90

Ожидаемый результат:

Ошибка: Автомобиль с номером 'C789DE90' не найден.

Кейс №2 «3D-моделирование и 3D-печать».

Задание 1

Постройте 3D-модель примитива согласно чертежу, представленному на рисунке 1. Экспортируйте детали в формат .stl.

Задание 2

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2. Экспортируйте детали в формат .stl.

Задание 3

Создайте сборку, имитирующую рациональное расположение и ориентацию деталей при печати на 3D-принтере на его рабочем столе. За поверхность рабочего стола принтера принимается базовая горизонтальная плоскость (Сверху. Тор, Вид сверху – в зависимости от выбранной САПР). Критерием рациональности выступает минимизация количества поддержек. Экспортируйте сборку в формат .stl.

Перв. примен.		Справ. №		Вариант 8		
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант 8 Примитив МАИ НИУ	
	Изм.	Лист	№ докум.	Подп.		Дата
	Разраб.					
	Пров.					
	Т. контр.					
	Н. контр.					
Утв.						
Лит.		Масса		Масштаб		
				1:5		
Лист		Листов 1				

Копировал

Формат А4

Рисунок 1

