**Лабораторная работа №9**

**Задание.**

Разработать программу работы со связанным списком, элементы которого состоят: из 8-символьного поля имени, однобайтового поля "возраста" и 4-байтового поля адреса следующего элемента списка. Элементы в списке должны быть упорядочены по алфавитному порядку хранимых в нем имен. Программа должна по запросам в диалоге с пользователем реализовывать следующие функции: вставлять в список задаваемый элемент (задаются значения полей имени и возраста), удалять из списка элемент с заданным именем, выводить список на экран. Память для размещения списка для упрощения построения программы зарезервировать в сегменте данных, задавая именованной константой максимальное число элементов списка в виде последовательности экземпляров структур для языка ассемблера или просто последовательности групп байтов.

**Ход работы:**

На рисунке 1 представлен пример работы программы, которая позволяет вводить из 8-символьного поля имени, однобайтового поля "возраста". Управление полем адреса этим полем адреса следующего элемента осуществляется автоматически, так как это ссылка на объект в Python

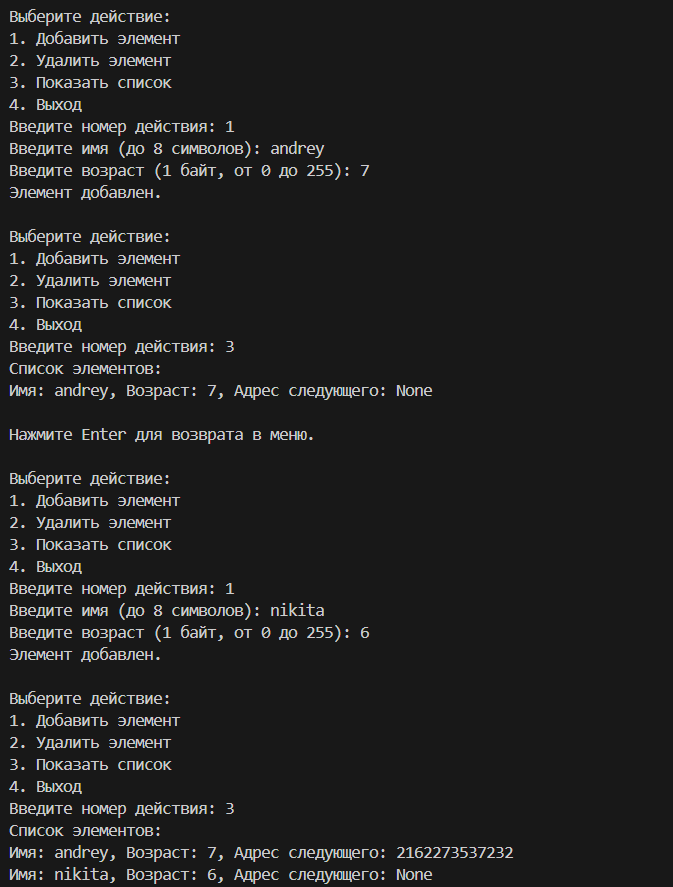
****

Рисунок 1

На рисунке 2 представлено удаление элемента из списка

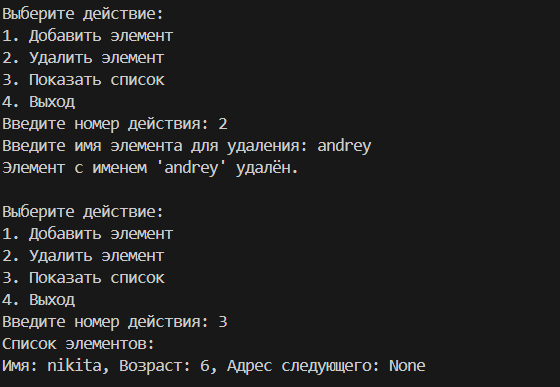


Рисунок 2

**Приложение**

class Node:

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

        # Инициализация узла списка

        # Поле имени обрезается или дополняется до длины 8 символов

        self.name = name.ljust(8)[:8]

        # Поле возраста (один байт, значение от 0 до 255)

        self.age = age

        # Указатель на следующий элемент (4-байтовое поле)

        self.next = None

    def \_\_repr\_\_(self):

        # Представление узла в виде строки для отладки

        next\_addr = id(self.next) if self.next else None  # Получаем адрес следующего узла или None

        return f"Node(name='{self.name.strip()}', age={self.age}, next={next\_addr})"

class LinkedList:

    def \_\_init\_\_(self):

        # Инициализация связанного списка, начально он пустой

        self.head = None

    def insert(self, name, age):

        # Вставка нового элемента в список с сохранением алфавитного порядка

        new\_node = Node(name, age)  # Создаем новый узел

        # Если список пуст или новый элемент меньше головы

        if self.head is None or new\_node.name < self.head.name:

            new\_node.next = self.head  # Новый узел указывает на текущую голову

            self.head = new\_node  # Новый узел становится головой списка

            return

        # Поиск места для вставки нового элемента

        current = self.head

        while current.next and current.next.name < new\_node.name:

            current = current.next  # Переходим к следующему узлу

        # Вставляем новый узел в найденное место

        new\_node.next = current.next  # Новый узел указывает на следующий элемент

        current.next = new\_node  # Текущий узел указывает на новый узел

    def delete(self, name):

        # Удаление элемента из списка по имени

        name = name.ljust(8)[:8]  # Обрезаем или дополняем имя до длины 8 символов

        # Проверяем, пуст ли список

        if self.head is None:

            print("Список пуст, удаление невозможно.")

            return

        # Если удаляемый элемент - это голова списка

        if self.head.name == name:

            self.head = self.head.next  # Голова списка сдвигается на следующий элемент

            print(f"Элемент с именем '{name.strip()}' удалён.")

            return

        # Поиск узла, предшествующего удаляемому

        current = self.head

        while current.next and current.next.name != name:

            current = current.next  # Переходим к следующему узлу

        # Если элемент не найден

        if current.next is None:

            print(f"Элемент с именем '{name.strip()}' не найден.")

            return

        # Удаляем элемент, перенаправляя указатель next

        current.next = current.next.next

        print(f"Элемент с именем '{name.strip()}' удалён.")

    def display(self):

        # Вывод списка на экран

        if self.head is None:

            print("Список пуст.")

            return

        # Проходим по всем элементам списка

        current = self.head

        print("Список элементов:")

        while current:

            # Получаем адрес следующего узла или None

            next\_addr = id(current.next) if current.next else None

            # Выводим информацию об узле

            print(

                f"Имя: {current.name.strip()}, Возраст: {current.age}, "

                f"Адрес следующего: {next\_addr}"

            )

            current = current.next  # Переходим к следующему узлу

        input("\nНажмите Enter для возврата в меню.")

if \_\_name\_\_ == "\_\_main\_\_":

    # Основная программа для взаимодействия с пользователем

    linked\_list = LinkedList()  # Создаем пустой связанный список

    while True:

        # Меню действий

        print("\nВыберите действие:")

        print("1. Добавить элемент")

        print("2. Удалить элемент")

        print("3. Показать список")

        print("4. Выход")

        choice = input("Введите номер действия: ")

        if choice == "1":

            # Добавление элемента в список

            name = input("Введите имя (до 8 символов): ")

            age = int(input("Введите возраст (1 байт, от 0 до 255): "))

            if 0 <= age <= 255:

                linked\_list.insert(name, age)  # Вставка элемента

                print("Элемент добавлен.")

            else:

                print("Возраст должен быть в диапазоне от 0 до 255.")

        elif choice == "2":

            # Удаление элемента из списка

            name = input("Введите имя элемента для удаления: ")

            linked\_list.delete(name)

        elif choice == "3":

            # Вывод списка

            linked\_list.display()

        elif choice == "4":

            # Выход из программы

            print("Выход из программы.")

            break

        else:

            # Неверный выбор

            print("Неверный выбор, попробуйте снова.")