Методические указания к лабораторной работе №2

Организация разветвленных и циклических вычислительных процессов

Целью лабораторной работы является изучение основных конструкции для организации разветвленных и циклических процессов. В рамках лабораторной работы слушатели научатся выбирать типы данных для эффективного решения задач, получат навыки создания программ, алгоритм работы которых определяется действиями пользователя.

Задание 1. Быстро и запятая

В нашем городе открывается новый ресторан быстрого питания «Быстро и запятая». В фиксированном меню ресторана: булочка, гамбургер, чизбургер, гранд, двойной гранд. Для приготовления каждого блюда требуется свой набор ингредиентов.

- Для булочки: 1 булочка с кунжутом.
- Для гамбургера: 1булочка и 1 котлета.
- Для чизбургера: 1 булочка, 1 котлета и 1 порция сыра.
- Для гранда: 1 булочка, 1 котлета, 2 порции сыра и 1 порция овощей.
- Для двойного гранда: 1 булочка, 2 котлеты, 2 порции сыра и 1 порция овощей.

Клиент может указать список своих предпочтений, но шеф приготовит из него только одно блюдо, то которое встречается в списке пользователя первым и на которое имеется необходимое количество продуктов.

Каждый день администратор проводит учет имеющихся продуктов и формирует список ингредиентов, состоящий из 4 целых чисел.

```
ingredients = [число_булочек, число_котлет, число_порций_сыра, число_порций _овощей]
# этот список создан в коде программы и имеет строго определенный порядок.
```

Напишите программу, позволяющую ввести список предпочтений клиента и выбрать заказ, который может быть выполнен. Если ни одно блюдо из текущих продуктов приготовить нельзя, то следует выдать сообщение «К сожалению, не можем предложить Вам бургер». Ресторан работает до последнего клиента (завершение программы только по согласию пользователя). При выдаче каждого очередного заказа следует корректировать список продуктов на кухне.

Пример диалога с пользователем. Пусть список ingredients = [2, 3, 2, 4].

Добрый день! У нас в меню
1. Булочка
2. Гамбургер
3. Чизбургер
4. Гранд

5. Двойной гранд

Введите через пробел не более 5 различных чисел от 1 до 5 в порядке убывания Ваших предпочтений к заказу.
1 3 4
Ваш заказ Булочка
Еще заказ (да/нет)? да
Введите через пробел не более 5 различных чисел от 1 до 5 в порядке убывания Ваших предпочтений к заказу.
5 3 1
Ваш заказ Двойной гранд
Еще заказ (да/нет)? да
Введите через пробел не более 5 различных чисел от 1 до 5 в порядке убывания Ваших предпочтений к заказу.
2 4 1
К сожалению, не можем предложить Вам бургер.
Еще заказ (да/нет)? нет
Спасибо за заказ!

Задание 2. Быки и коровы

Напишите программу, реализующую логическую игру «Быки и коровы». По правилам игры, компьютер случайным образом выбирает n-значное число таким образом, чтобы первой значащей цифрой не был 0, и чтобы все цифры, составляющие это число, были бы уникальными. Пользователь пытается угадать загаданное число, вводя предполагаемый ответ. Если в предложенном числе несколько цифр оказались на своем месте, компьютер сообщает, сколько есть таких цифр (они называются «быками»). Если в предложенном числе цифра совпадает с загаданной, но оказалась не на своем месте, компьютер сообщает, сколько есть таких цифр (это «коровы»).

После каждой попытки пользователь видит два целых числа – количество быков и коров через пробел.

Если число отгадано, то выводится сообщение о победе. В противном случае пользователю дается еще попытка или возможность завершить игру.

Это может пригодиться! В модуле random есть функция shuffle(), которая может перемешивать элементы списка. Если задать список, состоящий из цифр от 0 до 9, то можно получать случайное десятизначное число, где цифры не будут повторяться.

```
a = list(range(0,10))
random.shuffle(a)
```

Если первой цифрой сгенерировался 0, будем менять его местами со случайной цифрой, стоящей на позициях с 1 по 9.

```
if a[0] == 0:
    i = random.randint(1, 9)
    a[i], a[0] = a[0], a[i]
```

А этот код программы позволяет компьютеру сначала объединить массив в строку, а затем преобразовать в число. Например ['1', '2', '3', '4'] ⇒ '1234' ⇒ 1234.

```
int(''.join(map(str, a)))
```

Для отладки программы задайте число вручную, например, a = 1234.

Пример диалога с пользователем. Пусть загаданное число num = 1234.

```
Введите длину числа не более 10
4
Число загадано
Попробуете отгадать (да/нет)? да
5136
Число не отгадано
Быков 1 Коров 1
Попробуете отгадать (да/нет)? да
1534
Число не отгадано
Быков 3 Коров 0
Попробуете отгадать (да/нет)? нет
Будет скучно, возвращайтесь
```

Задание 3. Снова расписание

Напишите второй вариант программы «Мое расписание». Продолжаем улучшать наш будущий бот «Расписание».

Предварительная подготовка.

– Создать словарь, который хранит расписание в формате {день1: [занятие1, занятие2], день2: [занятие21, занятие22], …}.

Подзадачи, которые необходимо решить.

– Научиться добавлять в словарь занятие на заданный день.

Если в словаре еще нет расписания на данный день, добавить соответствующую пару «ключ: значение».

```
Schedule[day] = [subject]
```

Если расписания дня существует, используйте метод append.

– Научиться записывать расписание в файл и считывать его из файла (удобнее работать с файлом как с бинарным).

– Предусмотреть возможность получения списка занятий на заданный день.

– Научиться определять, в какой день указанное занятие.

Пример работы программ для перечисленных подзадач.

```
Введите день
пн
Введите занятия
БЖД
Добавлен новый день в расписании
{'пн': ['БЖД']}
Добавить еще занятие (да/нет?) да
```

```
Введите день
пн
Введите занятия
Математика
Добавлено занятие в существующий день
{'пн': ['БЖД', 'Математика']}

Сохранить файл (да/нет?) да
Файл сохранен под именем output
Вывод содержимого файла
{'пн': ['БЖД', 'Математика']}

Введите день
пн
БЖД
Математика

Введите занятие
Математика
День проведения занятия - пн
```