УДК 004.043

МАССИВЫ: ОТ ТЕОРИИ К ПРАКТИКЕ

Степанов В.М.

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники» филиал «Минский радиотехнический колледж», г. Минск, Республика Беларусь

Научный руководитель: Шумчик Ф.С. – заместитель директора по учебной работе, преподаватель высшей категории, канд. филол. наук, доцент

Аннотация. Массивы широко используются в программировании. Они обеспечивают удобный и эффективный способ управления данными. Благодаря им программист способен быстро и легко изменять огромное количество переменных, экономить своё время и ускорять работу программы.

Ключевые слова: массив, элементы, индекс, переменная, программирование игр.

Введение. Каждый знаком с массивами ещё со школы. Структуры данных для многих начинающих программистов, на которую чаще всего смотрят через призму скептицизма. Не понимают зачем, где и как это использовать, что толкает их без раздумий смело вычеркнуть все эти структуры из своего списка самых важных тем для изучения. Так давайте с вами рассмотрим, что такое массивы.

Массив – это структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целым) значения из некоторого заданного диапазона.

Для упрощения его можно представить в виде шкафчика или камеры хранения на вокзале или в магазине: набор ячеек, в каждой из которых может что-то лежать. Как и в шкафчике, каждая ячейка массива пронумерована, номер – это её индекс. Причём счёт идёт не от единицы, а от нуля [1].

Основная часть. Перед началом работы нужно выяснить, с каким массивом имеем дело. Различают самые разные размерности массивов. Это количество индексов, необходимое для конкретизации адресации элемента в массиве. Здесь выделяются следующие виды: одномерные, двумерные, многомерные.

Одномерный массив – это ряд пронумерованных элементов:

a = [1, 2, 3] print (a[2]) Окно вывода 3.

Двумерный массив можно представить в виде таблицы. Чтобы обратиться к элементу, нужно ввести его строку и номер ячейки в строке:

A=[[1, 2, 3], [4, 5, 6],[7, 8, 9]]print(A[2][0])Окно вывода 7

Отметим основные базовые операции, проводимые над массивами:

1) обход, то есть вывод одного за другим всех элементов массива;

2) вставка, то есть добавление элемента с заданным индексом;

3) удаление элемента с заданным индексом;

4) поиск элемента по заданному индексу или значению;

5) изменение элемента с заданным индексом.

Рассмотрим данные операции.

1. Обход.

Обход и вывод элементов массива происходит следующим образом:

a = [1, 2, 3, 4, 5]print(' '.join([str(i) for i in a])) Окно вывода 1 2 3 4 5

2. Вставка.

Эта операция заключается в добавлении в массив одного или нескольких элементов данных. При этом они добавляются в начало, конец массива или любое его место с заданным индексом:

a = [1, 2, 3, 4, 5] a.insert(2, 15) print(' '.join([str(i) for i in a])) Окно вывода 1 2 15 3 4 5

3. Удаление.

Эта операция заключается в удалении элемента из массива и реорганизации оставшихся элементов:

a = [1, 2, 3, 4, 5] a.pop(1) print(' '.join([str(i) for i in a])) Окно вывода 1 3 4 5

4. Поиск.

Поиск элемента массива выполняется по его значению или индексу:

a = [1, 2, 3, 4, 5] print ('Элемент находится на позиции ', a.index(4))Окно вывода Элемент находится на позиции 3

5. Изменение.

Эта операция заключается в изменении элемента массива с заданным индексом:

a = [1, 2, 3, 4, 5] a[2] = 10 print (' '.join([str(i) for i in a])) Окно вывода 1 2 10 4 5 [2]

Надобность массивов возникает при работе с большим количеством переменных. Неужели нам придётся вручную обрабатывать 1000 элементов? Тут нам массивы и помогут.

Представим ситуацию, что мы пишем программу, в которой будет храниться температура окружающей среды по дням за декабрь. Если не использовать массивы, то для каждого значения температуры придётся создавать свою переменную. Это информация будет храниться разрозненно, и переменные никак не связаны друг с другом. Если мы захотим посчитать среднюю температуру за декабрь, придётся писать длинную формулу:

SrDec = (1dec + 2dec + 3dec + … + 31dec)/31

А если вдруг окажется, что термометр даёт ошибку в +3 градуса? Чтобы исправить ошибку, придётся вручную отнимать от каждой переменной 3 – и так для каждой из 31 строки кода.

Или, допустим, мы захотим учитывать температуру не только в нашем городе, но и в соседнем. Тогда в названии каждой переменной нужно будет дополнительно указывать не только дату, но и название города – так и запутаться можно. Гораздо проще пойти другим путём: создать массив «Декабрь» и записывать температуру уже внутри него:

```
dec_temperatures = []
for i in range(1, 32):temp = float(input(f"Введите температуру на день {i}: ")
dec_temperatures.append(temp)
print(dec_temperatures)
```

Программа выведет информацию о температуре за декабрь.

Так программа понимает, что эти элементы связаны между собой**.** Теперь мы за пару строчек кода можем найти среднемесячную температуру – достаточно сказать программе: сложи все элементы и раздели их на длину массива. Или прибавить три градуса (сказать: прибавь 3 к каждому элементу):

```
1) avg_temp = sum(dec_temperatures) / len(dec_temperatures)
print(f" Средняя температура за декабрь составляет: {avg_temp:.2f}").
```

Если добавить пару строк, то программа выведет среднюю температуру.

```
2) for i in range(len(dec_temperatures)):
dec_temperatures[i] += 3
print(dec_temperatures).
```

Здесь показано, как увеличить каждое значение на 3, например, если прибор ошибся.

Если захотим измерять температуру в другом городе, можно просто создать другой массив – и их значения точно не перемешаются. При этом обратиться к показателям температуры любого города легче лёгкого, ведь каждый элемент имеет свой индекс, как номер ячейки в камере хранения. Нужно лишь запомнить, что нумерация обычно идёт от нуля. Данные хранятся более упорядоченно, нет никакого нагромождения из десятков и сотен переменных, которые нужно держать в голове, чтобы не запутаться и не ошибиться [3].

Массивы нашли применение в программировании игр.

Во многих играх необходимо отслеживать большое количество объектов: враги, бонусы, препятствия и т.д. Массивы можно использовать для хранения этих объектов, что упрощает управление ими и их обновление во время игры. Например, разработчик игры может создать массив вражеских объектов, каждый со своим набором свойств – здоровье, скорость, сила атаки и т.д. Это позволяет быстро и эффективно отслеживать всех врагов в игре, упрощая создание увлекательных и сложных уровней.

Приведем несколько примеров использования массивов в играх.

1. Хранение данных об игроках. Во многих играх необходимо хранить данные об игроках и управлять ими. Например, игре может потребоваться отслеживать здоровье, инвентарь и положение игрока. Использование массива для хранения этих данных может упростить управление и обновление:

player_datplayer_data = ["John", 100, ["sword", "shield"], (0,0)]

player_data[1] выведет 100

player_data[2][1] выведет *shield*

Таким образом, у нас есть массив, в котором храниться имя персонажа *John*, его уровень *100*, у него в инвентаре *sword and shield*, и его позиция *(0,0)*. Это крайне удобно использовать для вывода данных на игровой экран.

2. Управление игровыми объектами. В играх часто используется множество объектов, которыми необходимо управлять. Например, в игре могут быть враги, бонусы и препятствия, которые необходимо отслеживать. Массив можно использовать для хранения информации о каждом объекте – его расположение, тип и поведение:

```
enemies = [
  {"type": "zombie", "location": [10, 10], "health": 100},
  {"type": "skeleton", "location": [20, 20], "health": 50},
  {"type": "ghost", "location": [30, 30], "health": 200}
]
```

enemies[1]["type"] выведет *skeleton*

enemies[2]["health"] выведет 200

Здесь у нас есть типы врагов, их место положения и их здоровье.

3. Сохранение игровых уровней. Во многих играх есть уровни, которыми необходимо управлять. На каждом уровне могут быть разные объекты, враги и испытания. Массив можно использовать для хранения информации о каждом уровне – расположение объектов и врагов:

```
levels = [
  {"name": "level1", "enemies": ["zombie", "skeleton"], "objects": ["rock", "tree"]},
  {"name": "level2", "enemies": ["ghost", "goblin"], "objects": ["mushroom", "bush"]}
]
```

levels[0]["objects"][1] выведет *tree*

levels[1]["enemies"][0] выведет *ghost*

Мы сохранили локации, какие на них находятся монстры и объекты.

4. Сохранение позиций игроков. Во многих играх позиции игроков отслеживаются и обновляются на протяжении всей игры. Массив можно использовать для хранения координат x и y каждого игрока, что обеспечивает эффективный доступ и управление позициями игроков:

player_positions = [[0, 0], [5, 10], [10, 20]]

player_positions[0] выведет [0, 0]

player_positions[1][0] выведет 5

Благодаря этому мы можем сохранять позиции игроков. И, например, при возрождении, обращаясь к массиву, персонаж будет появляться на сохранённой позиции (координатах).

5. Отслеживание игровых результатов: игровые очки можно отслеживать с помощью массива, где каждый элемент соответствует счету игрока. По мере того, как игроки зарабатывают очки, их счет может обновляться в массиве:

player_scores = [0, 0, 0]

player_scores[1] += 10

6. Хранение игровых объектов. Игровые объекты – враги и бонусы – можно хранить в массиве для эффективного доступа и управления ими. Каждый элемент массива может соответствовать другому объекту с такими свойствами, как положение, здоровье и скорость:

```
enemies = [
  {'position': [5, 5], 'health': 10, 'speed': 2},
  {'position': [10, 10], 'health': 5, 'speed': 5},
```

```
  {'position': [15, 20], 'health': 20, 'speed': 1}
]
enemies[0]['position'] выведет [5, 5]
enemies[1]['health'] выведет 5
```

Заключение. В целом, массивы – это мощный инструмент для разработчиков игр, позволяющий управлять данными в играх и манипулировать ими. Их можно использовать для хранения информации об игроках, игровых объектах, уровнях и игровых состояниях. Используя массивы, разработчики игр могут создавать эффективные, организованные и динамичные игры [4].

Массивы широко используются в программировании, поскольку они обеспечивают удобный и эффективный способ управления коллекциями данных. Благодаря им программист способен быстро и легко изменять огромное количество переменных, экономить своё время и ускорять работу программы. Сложно представить, как бы выглядело программирование без использования массивов.

Список литературы

1. *https://skillbox.ru/media/code/chto-takoe-massiv-i-kak-on-ustroen/.*
2. *https://otus.ru/journal/massivy-v-programmirovanii-opisanie-i-napolnenie-dannymi/.*
3. *https://www.youtube.com/watch?v=47_LhSf-ago&t=158s.*
4. *https://www.youtube.com/watch?v=aHPdNVtqRoI&t=187s.*

UDC 004.043

ARRAYS: FROM THEORY TO PRACTICE

Stepanov V.M.

Belarusian State University of Informatics and Radioelectronics affiliate
Minsk Radioengineering College, Minsk, Republic of Belarus

Shumchik F.S. – Deputy Director for Academic Affairs, teacher of the highest category, PhD, associate professor

Annotation. Arrays are widely used in programming. They provide a convenient and efficient way to manage data. Thanks to them, the programmer is able to quickly and easily change a huge number of variables, save his time and speed up the program.

Keywords: array, elements, index, variable, game programming.