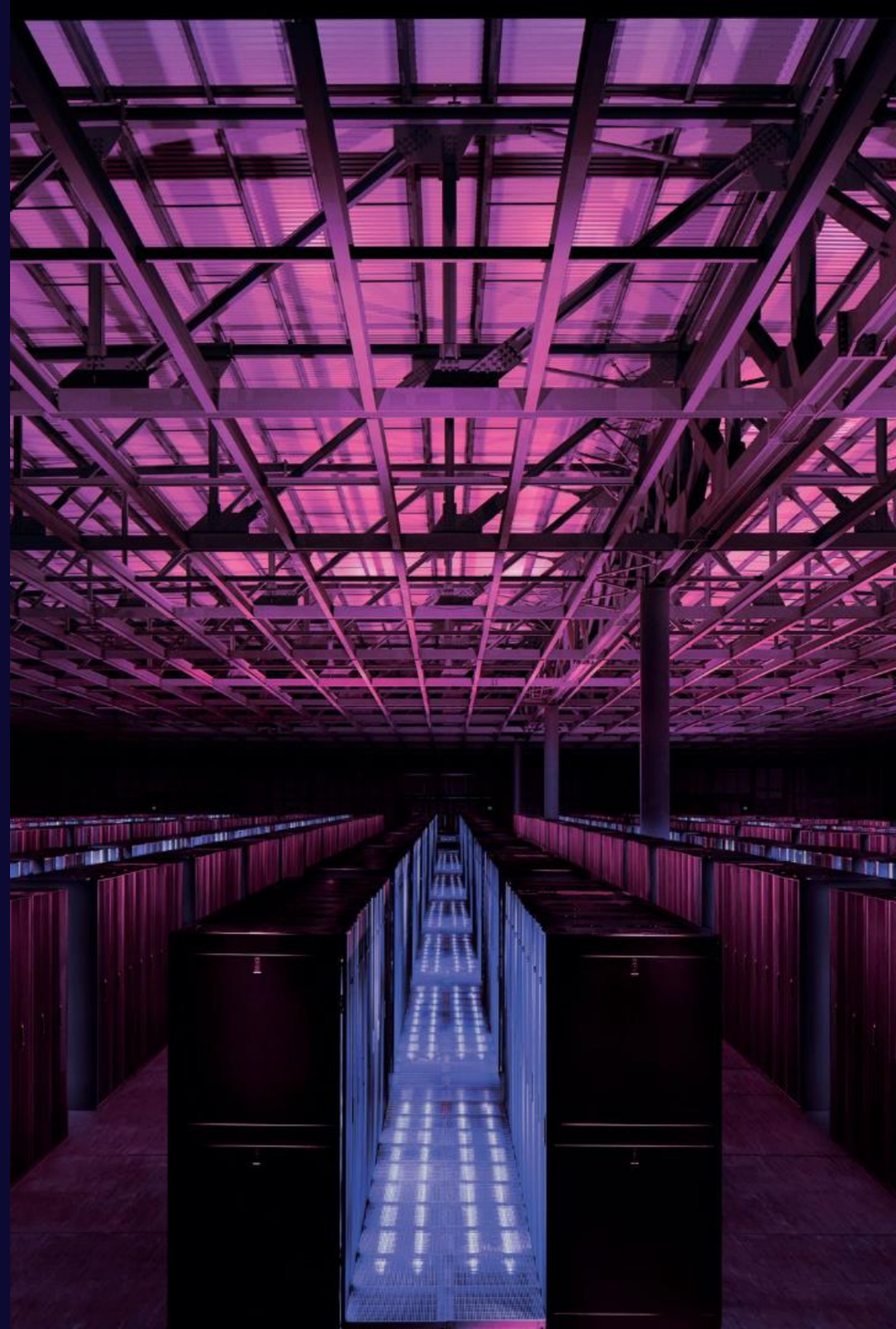
Рассмотрим несколько структур данных, реализованных в Julia

В этой презентации мы познакомимся с различными структурами данных, используемыми в языке программирования Julia, и увидим, как они помогают нам работать с данными эффективно и гибко.



Структуры данных в Julia

Списки

Упорядоченные коллекции элементов, где каждый элемент имеет индекс для доступа.

Массивы

Объекты, содержащие фиксированное количество элементов одного типа.

Словари

Коллекции пар "ключ-значение", где каждому ключу соответствует значение.

Множества

Неупорядоченные коллекции уникальных элементов без дубликатов.

Обзор нескольких структур данных

```
In [12]: phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),"Бухгалтерия" =>"555-2368")
         keys(phonebook)
```

```
KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."
```

```
In [13]: values(phonebook)
```

```
ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")
```

```
In [14]: pairs(phonebook)
```

```
Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5309", "333-5544")
```

```
In [15]: haskey(phonebook,"Иванов И.И.")
```

```
true
```

```
In [16]: phonebook["Сидоров П.С."]="555-3344"
```

```
"555-3344"
```

```
In [17]: pop!(phonebook, "Иванов И.И.")
```

```
In [17]: pop!(phonebook, "Иванов И.И.")
```

```
("867-5309", "333-5544")
```

```
In [18]: a= Dict("foo"=>0.0,"bar"=>42.0);
         b= Dict("baz"=>17, "bar"=>13.0);
         merge(a,b),merge(b,a)
```

```
(Dict{String, Real}("bar" => 13.0, "baz" => 17
```

```
In [20]: A=Set([1,3,4,5])
         B=Set("abrakadabra")

         S1=Set([1,2])
         S2=Set([3,4])
         issetequal(S1,S2)
```

```
false
```

```
In [23]: S3=Set([1,2,2,3,1,2,3,2,1]);
         S4=Set([2,3,1])
         issetequal(S3,S4)
```

```
true
```

```
In [24]: C=union(S1,S2)
```

```
Set{Int64} with 4 elements:
```

```
In [37]: c=rand(2,3)
```

```
2×3 Matrix{Float64}:
 0.32506   0.989055  0.622885
 0.816279  0.175961  0.757636
```

```
In [39]: roots=[sqrt(i) for i in 1:10]
```

```
10-element Vector{Float64}:
 1.0
 1.4142135623730951
 1.7320508075688772
 2.0
 2.23606797749979
 2.449489742783178
 2.6457513110645907
 2.8284271247461903
 3.0
 3.1622776601683795
```

```
In [40]: ar_1=[3*i^2 for i in 1:2:9]
```

```
5-element Vector{Int64}:
   3
  27
  75
 147
 243
```

```
In [42]: ar_2=[i^2 for i=1:10 if(i^2%5!=0&&i^2%4!=0)]
```

```
4-element Vector{Int64}:
  1
  9
 49
```

Общие функции (методы) для структур данных

1

Добавление

Функции для добавления новых элементов в структуру данных.

2

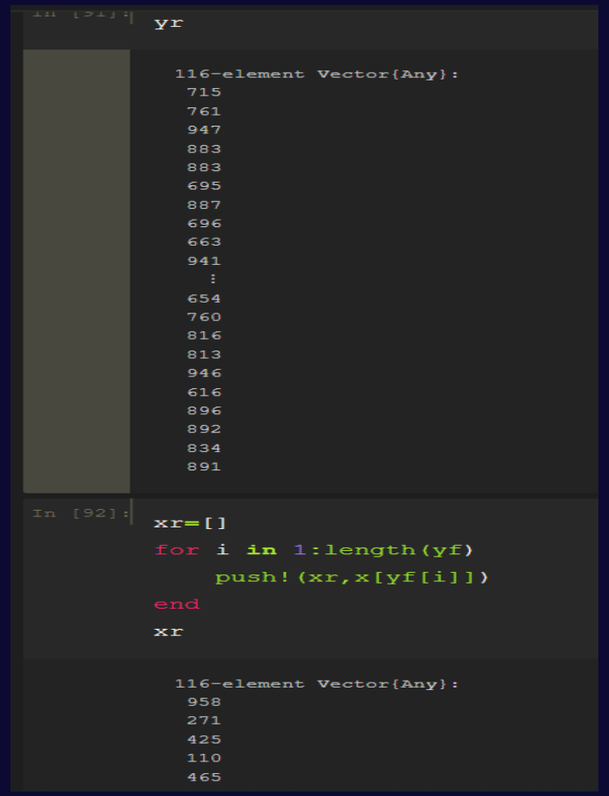
Удаление

Функции для удаления элементов из структуры данных.

3

Поиск

Функции для поиска определенных элементов или позиций в структуре данных.



```
In [12]: phonebook = Dict("Иванов И.И." => ("867-5309","333-5544"),
         keys(phonebook)

KeySet for a Dict{String, Any} with 2 entries. Keys:
  "Бухгалтерия"
  "Иванов И.И."

In [13]: values(phonebook)

ValueIterator for a Dict{String, Any} with 2 entries. Values:
  "555-2368"
  ("867-5309", "333-5544")

In [14]: pairs(phonebook)

Dict{String, Any} with 2 entries:
  "Бухгалтерия" => "555-2368"
  "Иванов И.И." => ("867-5309", "333-5544")

In [15]: haskey(phonebook,"Иванов И.И.")

true

In [16]: phonebook["Сидоров П.С."]="555-3344"

"555-3344"

In [17]: pop!(phonebook, "Иванов И.И.")
```

Преимущества использования структур данных

1 Эффективность

Структуры данных позволяют эффективно организовывать и обрабатывать большие объемы данных.

2 Гибкость

Структуры данных предоставляют гибкие методы доступа и изменения данных.

3 Удобство

Структуры данных облегчают работу с данными и повышают понимание их организации.

Заключение

Структуры данных являются фундаментальными для эффективной работы с данными в программировании. Познакомившись с различными структурами данных в Julia, мы обрели мощный инструмент для решения разнообразных задач.