Arrays are indexed sets.

Pairs and Maps

- Let A and B be sets. The Cartesian product of A and B, denoted by $A \times B$, is the set of all ordered pairs (a, b) where $a \in A$ and $b \in B$:

$$A \times B = \{ (a,b) \mid a \in A \text{ and } b \in B \}$$

- A map is an associative container, whose elements are key-value pairs. The key serves as an index into the map, and the value represents the data being stored and retrieved.

Associative Array (Dictionaries)

- An associate array is a map in which elements are indexed by a key rather than by their position.

$$a[i] = \begin{cases} v, & \text{if } i \mapsto v \text{ in } a \\ \perp, & \text{otherwise} \end{cases}$$

- Example:

$a = \{ (\text{“u”} \mapsto 345), (\text{“v”} \mapsto 2), (\text{“w”} \mapsto 39), (\text{“x”} \mapsto 5) \}$

$a[\text{“w”}] = 39$

$a[\text{“z”}] = \perp$

From Indices to Keys

- We can define an adapter class that defines an indexer:

ArrayIndexer.h — Sets

```
#include <string>

class IntArrayIndexer
{
private:

    const int* fArray;
    size_t fSize;

public:

    IntArrayIndexer( const int aArray[], size_t aSize );

    const size_t size() const;
    const int& operator[]( const size_t aIndex ) const;
    const int& operator[]( const std::string& aKey ) const;
};
```

Line: 1 C++ Tab Size: 4

Indexer Constructor

Arrays are passed as pointers to the first element to functions in C++.

```cpp
#include "ArrayIndexer.h"

#include <stdexcept>

using namespace std;

IntArrayIndexer::IntArrayIndexer( const int aArray[], size_t aSize ) :
    fArray(aArray),
    fSize(aSize)
{}
```

Line: 1 C++ Tab Size: 4

We must use member initializer to initialize const instance variables!

Basic Indexer Operations

ArrayIndexer.cpp — Sets

```
15  const size_t IntArrayIndexer::size() const
16  {
17      return fSize;
18  }
19
20  const int& IntArrayIndexer::operator[]( const size_t aIndex ) const
21  {
22      if ( aIndex < fSize )
23      {
24          return fArray[aIndex];
25      }
26      else
27      {
28          throw out_of_range( "Illegal array index." );
29      }
30  }
31
```

One-sized range test.

Line: 1 C++ Tab Size: 4

The Indexer

ArrayIndexer.cpp — Sets

```
const int& IntArrayIndexer::operator[]( const std::string& aKey ) const
{
    size_t lIndex = 0;

    for ( size_t i = 0; i < aKey.size(); i++ )
    {
        lIndex = lIndex * 10 + (aKey[i] - '0');
    }

    return (*this)[lIndex];
}
```

stoi

forward to [](size_t)

Line: 1 C++ Tab Size: 4

- We use the const specifier to indicate that the operator[]:
  - is a read-only getter
  - does not alter the elements of the underlying collection
- We use a const reference to avoid copying the original value stored in the underlying collection.

Testing the Indexer

COS3008

```
Kamala:COS3008 Markus$ ./ArrayIndexer
Indexed sum of [1,2,3,4,5] is 15
Kamala:COS3008 Markus$ _
```

Main.cpp — Sets

```cpp
int main()
{
    int lArray[] = { 1, 2, 3, 4, 5 };
    IntArrayIndexer lIndexer( lArray, sizeof( lArray ) / sizeof( int ) );

    string lKeys[] = { "0", "1", "2", "3", "4" };
    int lSum = 0;

    for ( size_t i = 0; i < lIndexer.size(); i++ )
    {
        lSum += lIndexer[lKeys[i]];
    }

    cout << "Indexed sum of [1,2,3,4,5] is " << lSum << endl;

    return 0;
}
```

Line: 1 C++ Tab Size: 4

How can we define an indexer in Java?

The Transition to Java

- We need to define an Indexer class.
- Java does not support operator overloading. So, we need to map [] to a member function.
- The built-in type Integer provides the required conversion operations.
- We use IndexOutOfBoundsException to signal an index error.

Indexer's at(String aKey) Method

Indexer.java

```java
public class Indexer
{
    private int[] fArrayElements;

    public Indexer( int[] aArray )
    {
        fArrayElements = aArray;
    }

    // Indexer behavior
    public int at( String aKey )
    {
        int lIndex = (new Integer( aKey )).intValue();

        if ( lIndex < fArrayElements.length )
            return fArrayElements[lIndex];
        else
            throw new IndexOutOfBoundsException( "Index out of bounds!" );
    }

    public static void main( String[] args ) { ... }
}
```

Line: 24 Column: 1 Java Tab Size: 4 main(String[] args)

The Indexer's main Method

Indexer.java

```java

public class Indexer
{

    ...

    public static void main( String[] args )
    {
        int[] a = { 1, 2, 3, 4, 5 };
        Indexer indexer = new Indexer( a );
        String[] keys = { "0", "1", "2", "3", "4" };
        int Sum = 0;

        for ( int i = 0; i < 5; i++ )
          Sum += indexer.at( keys[i] );

        System.out.println( "Indexed sum of [1,2,3,4,5] is " + Sum );
    }
}

```

Line: 20 Column: 1 Java Tab Size: 4 main(String[] args)