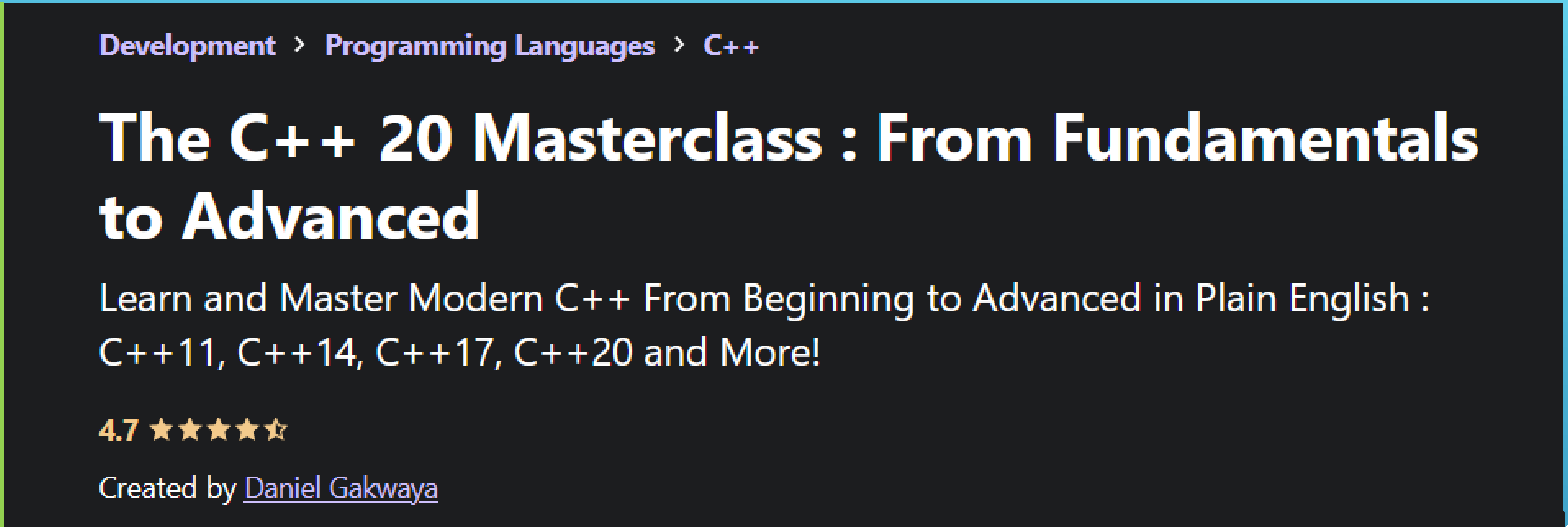


Development › Programming Languages › C++

The C++ 20 Masterclass : From Fundamentals to Advanced

Learn and Master Modern C++ From Beginning to Advanced in Plain English : C++11, C++14, C++17, C++20 and More!

4.7 ★★★★☆

Created by Daniel Gakwaya

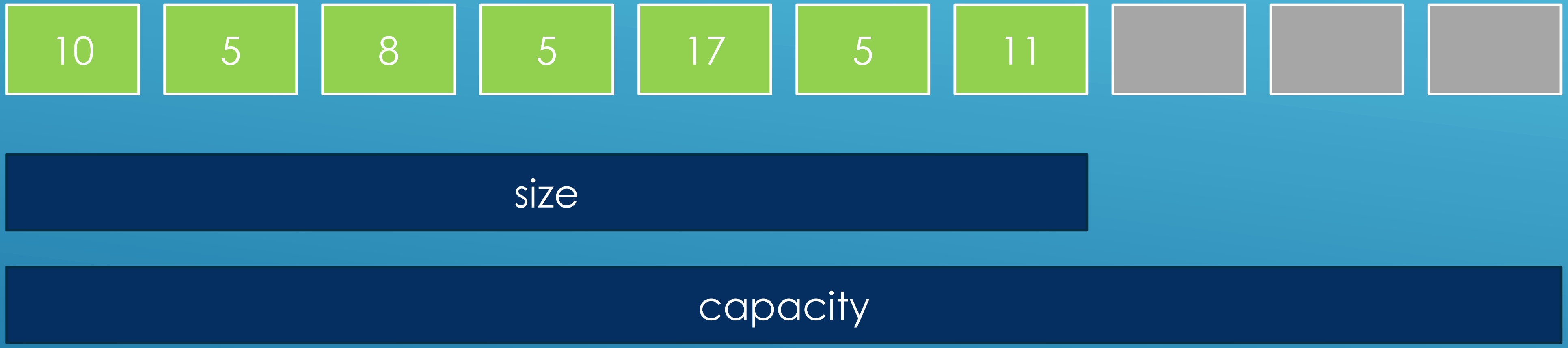
Slides

Section : Box Container class – Practicing what we know

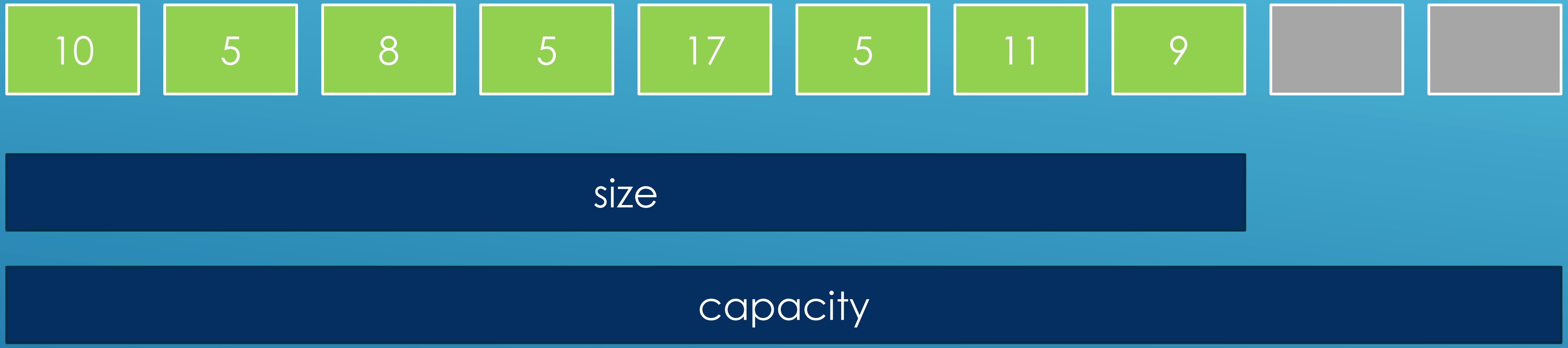
Slide intentionally left empty

BoxContainer : Introduction

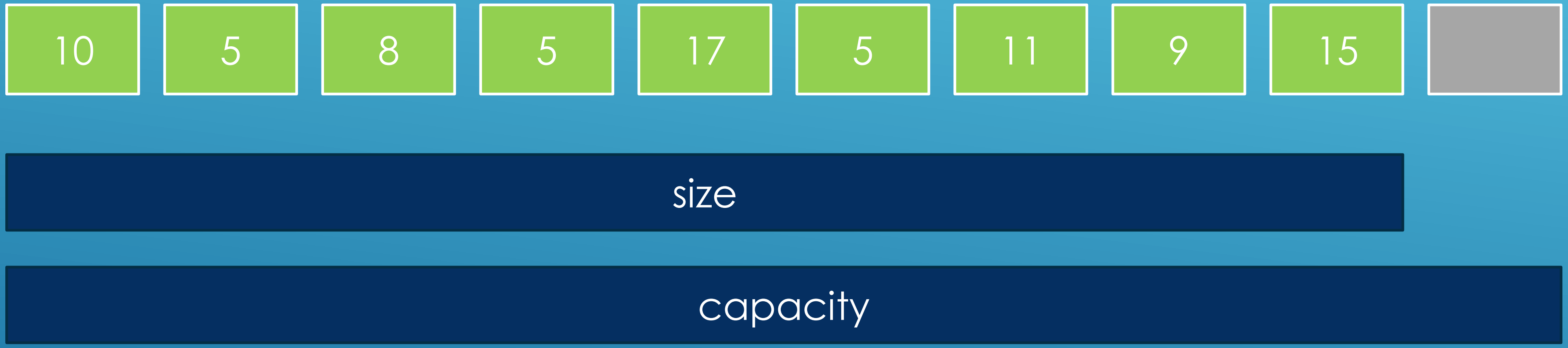
```
. BoxContainer is a container class that provides the features
        . add_item
        . remove_item
        . remove_all

. We can add BoxContainer's up with :
        . +=
        . +

. Additionaly we can :
        . Stream insert BoxContainers with opreator<<
        . Copy construct BoxContainer's
        . Copy assign BoxContainer's
```


10
5
8
5
17
5
11
size
capacity

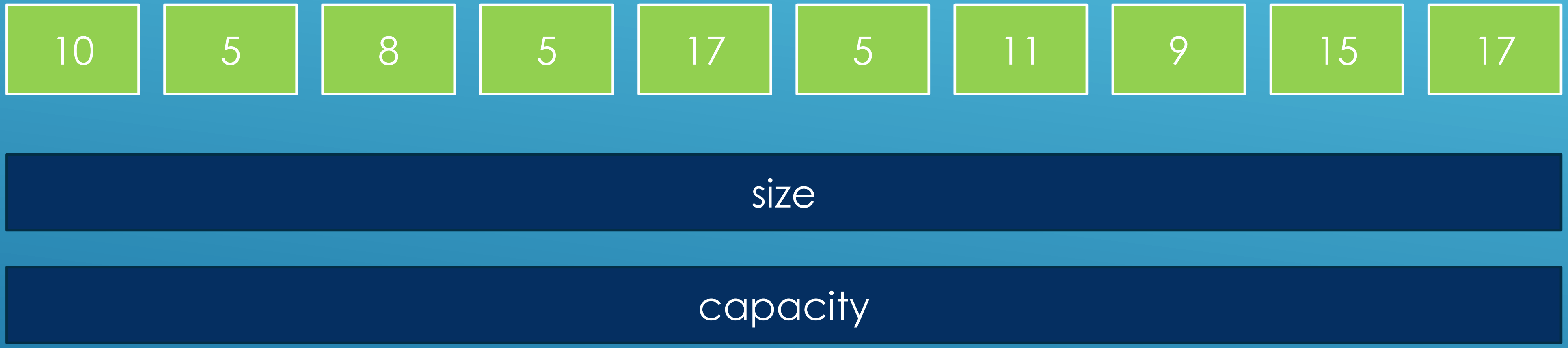
Adding elements



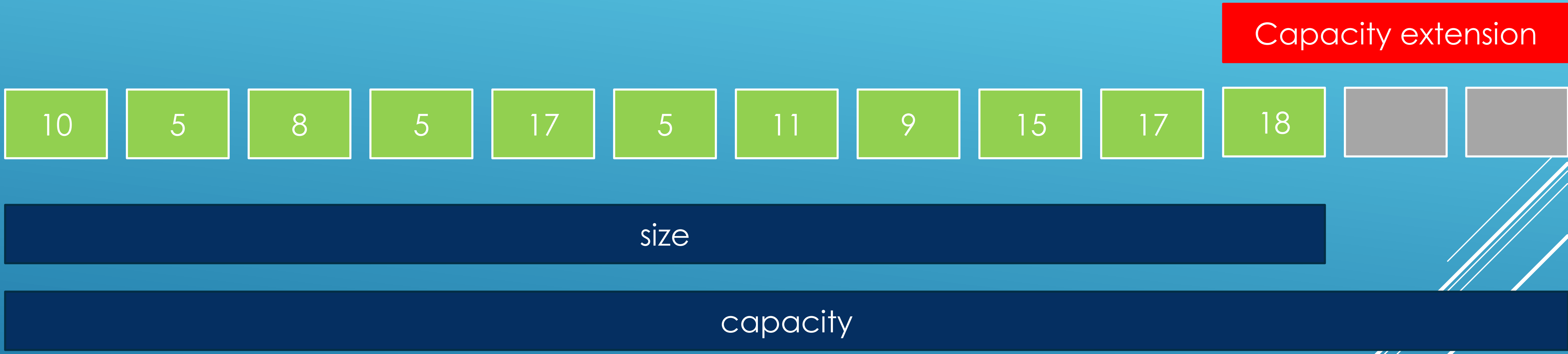
Adding elements



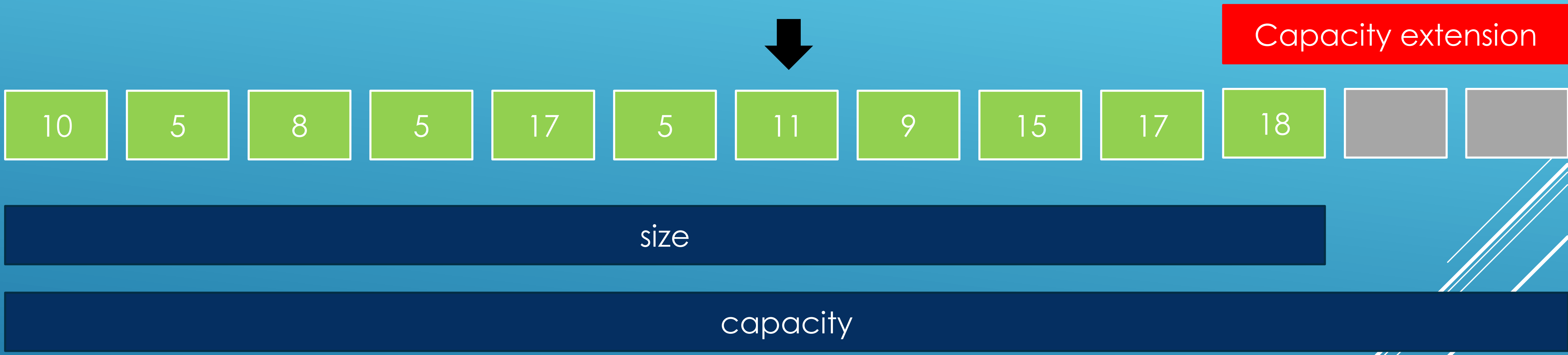
Adding elements



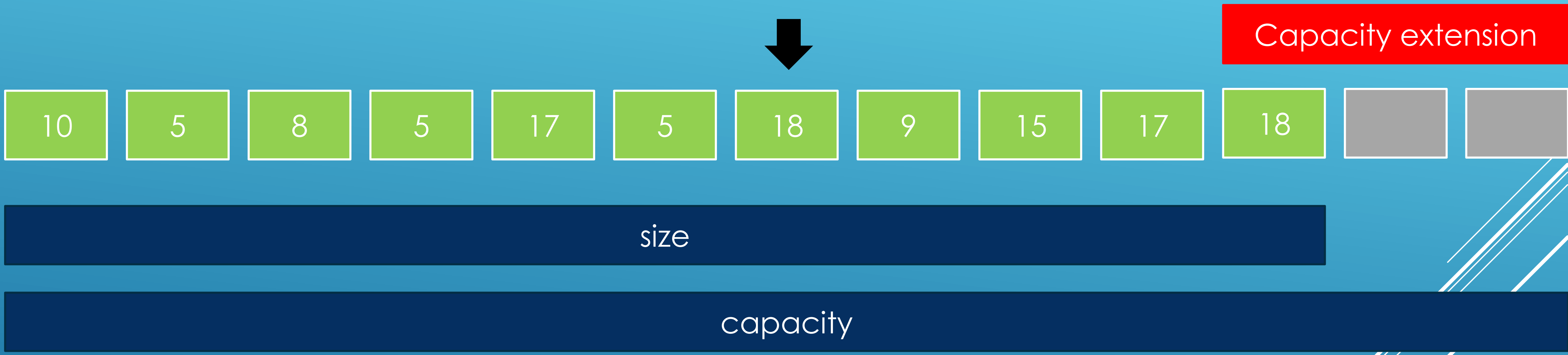
Adding elements



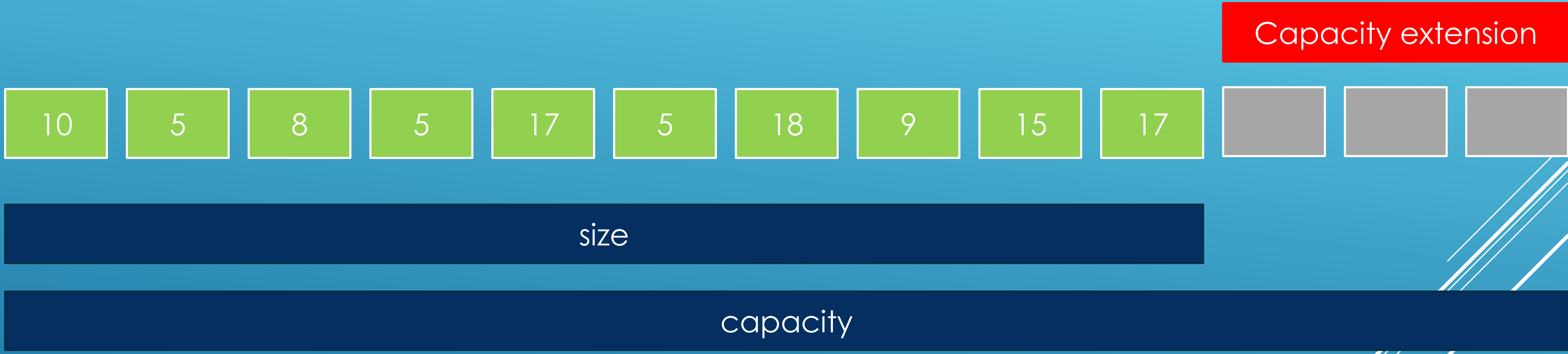
Removing elements



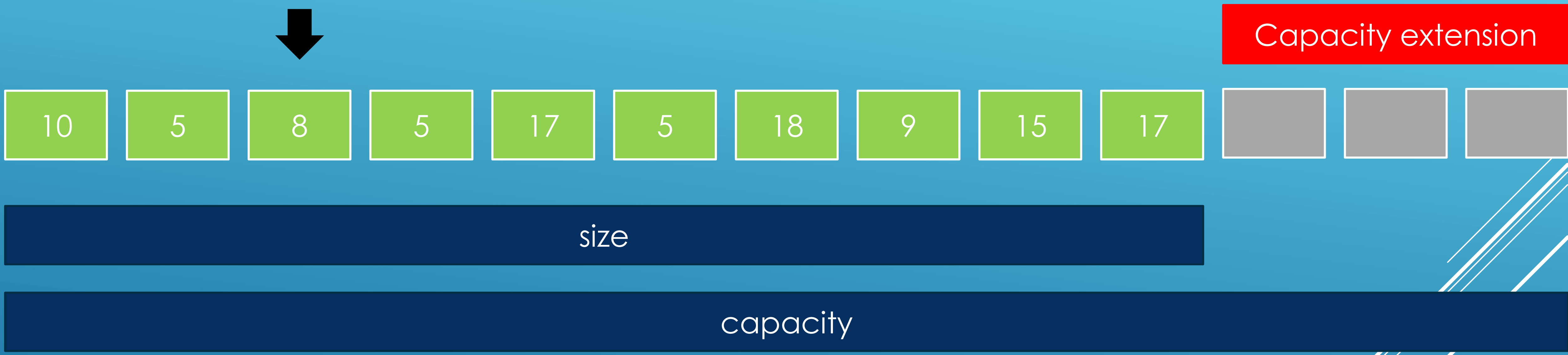
Removing elements



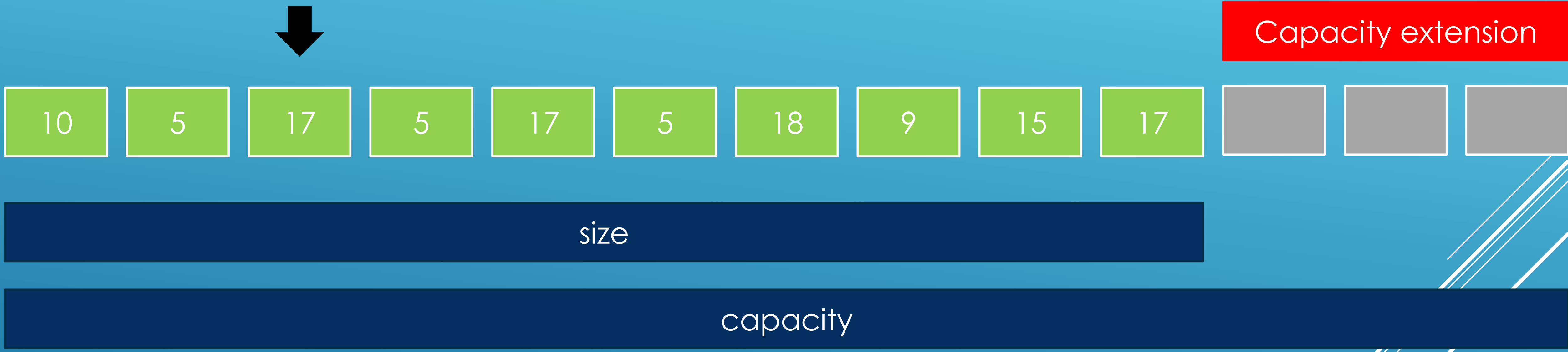
Removing elements



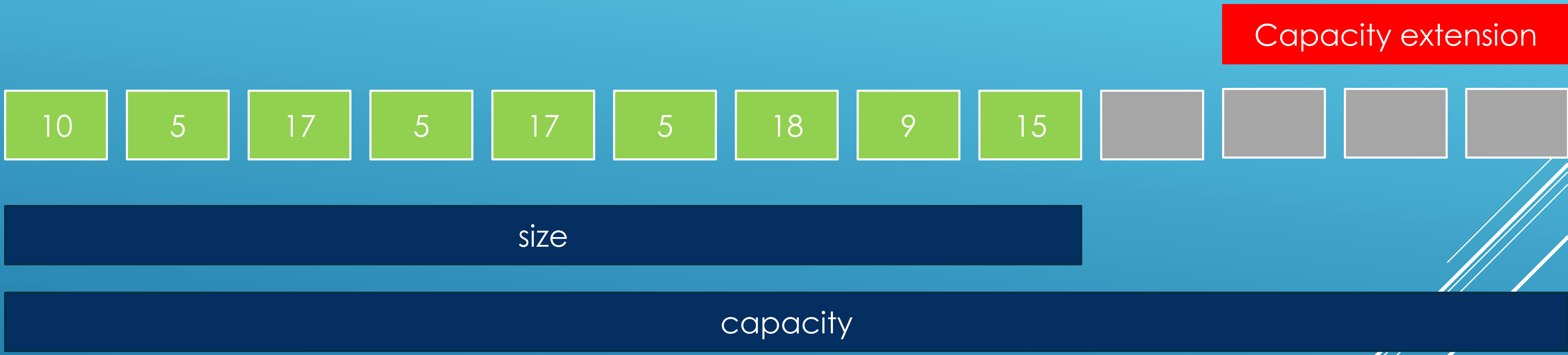
Removing elements



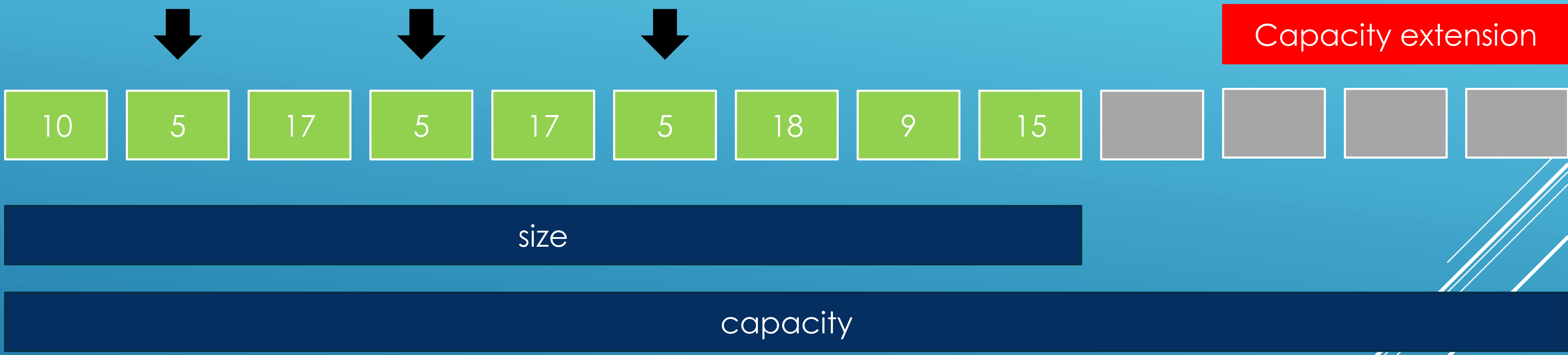
Removing elements



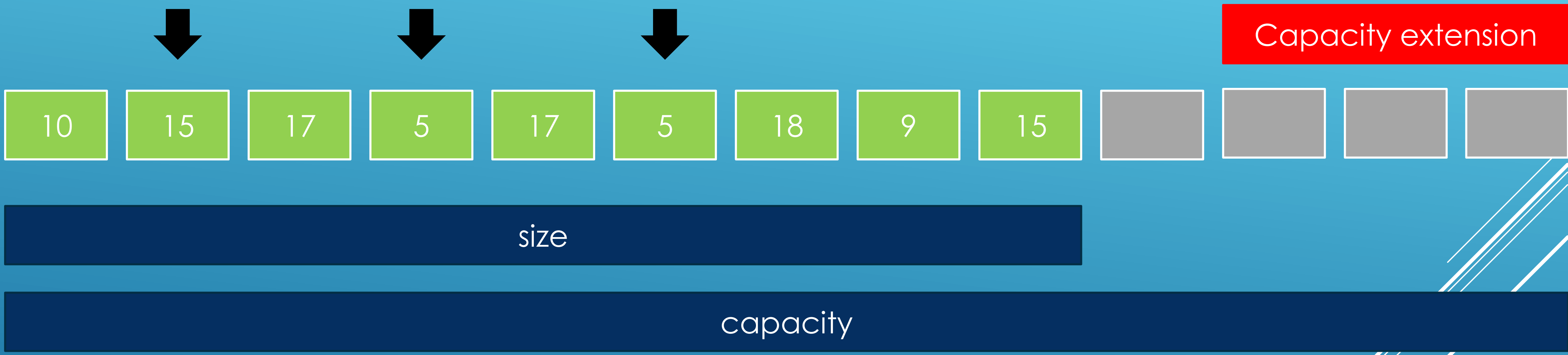
Removing elements



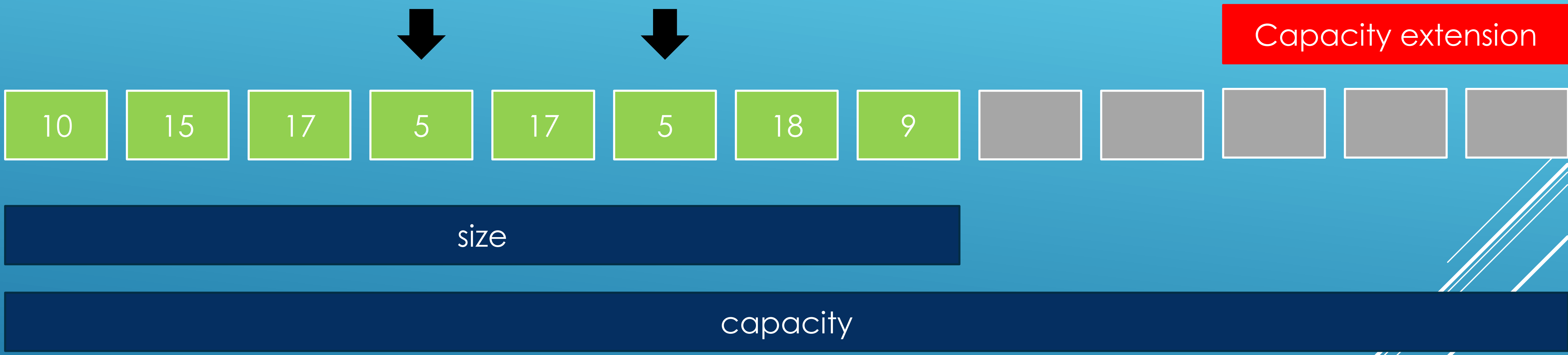
Removing elements

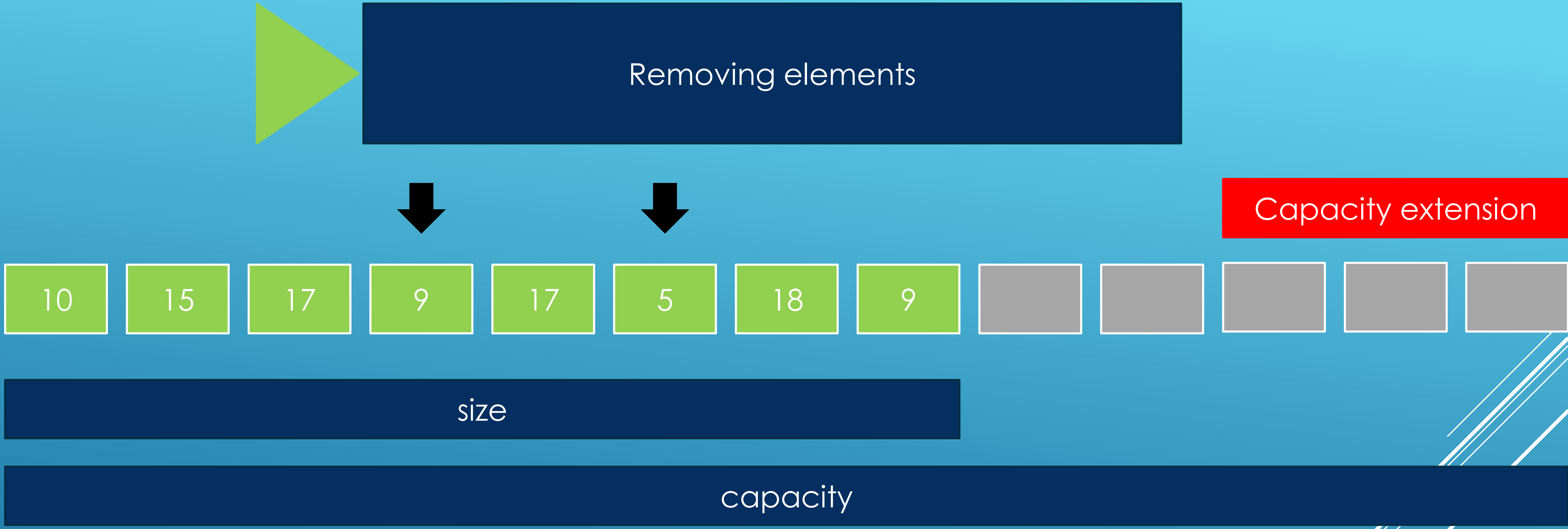


Removing elements

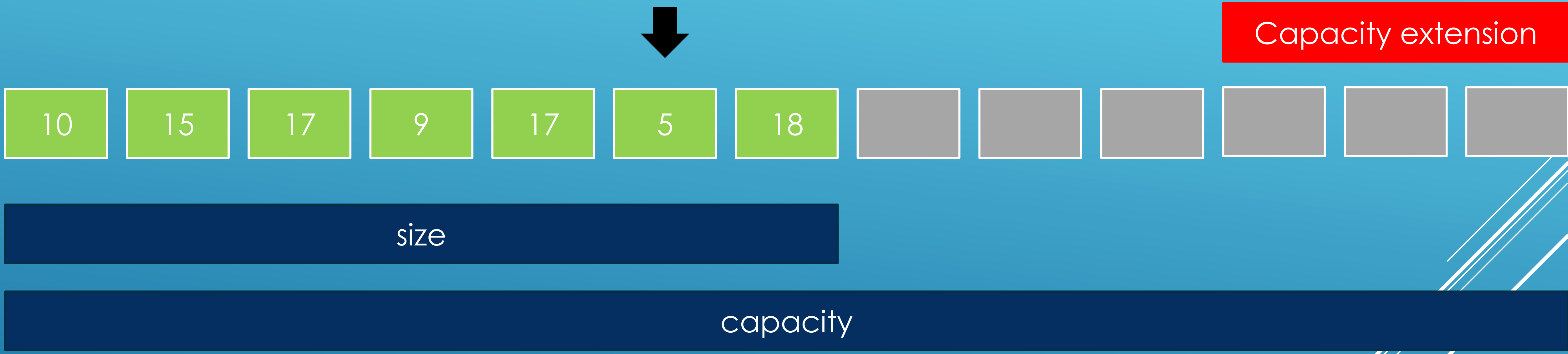


Removing elements

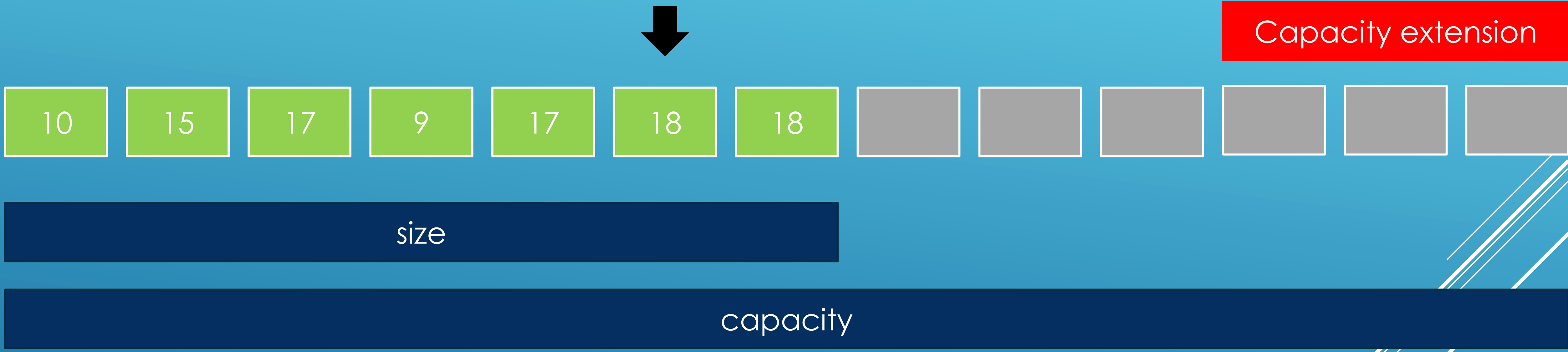



Removing elements
Capacity extension
10
15
17
9
17
5
18
9
size
capacity

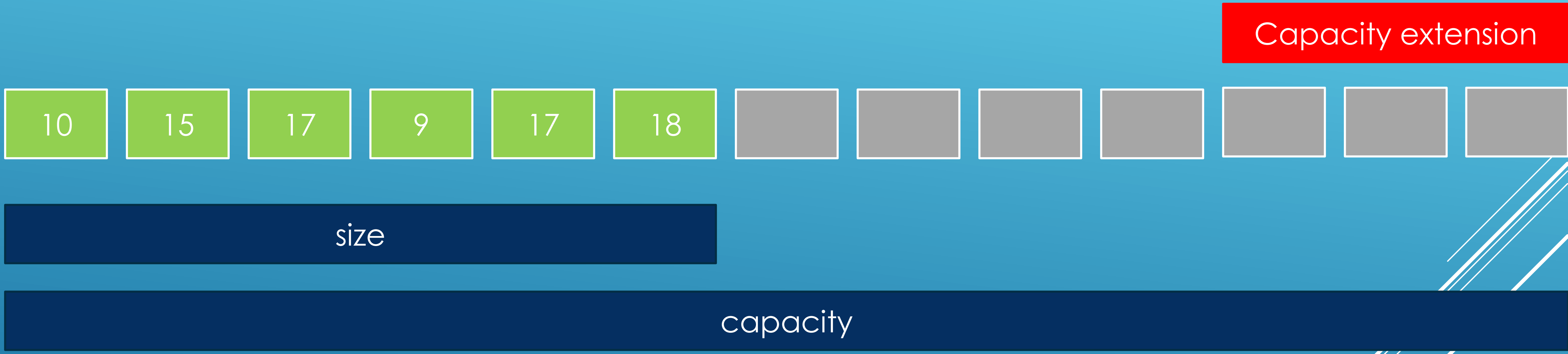
Removing elements



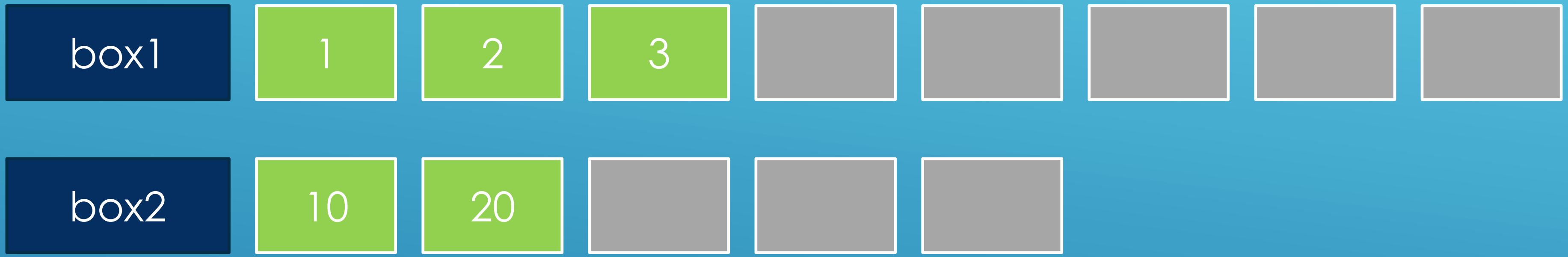
Removing elements



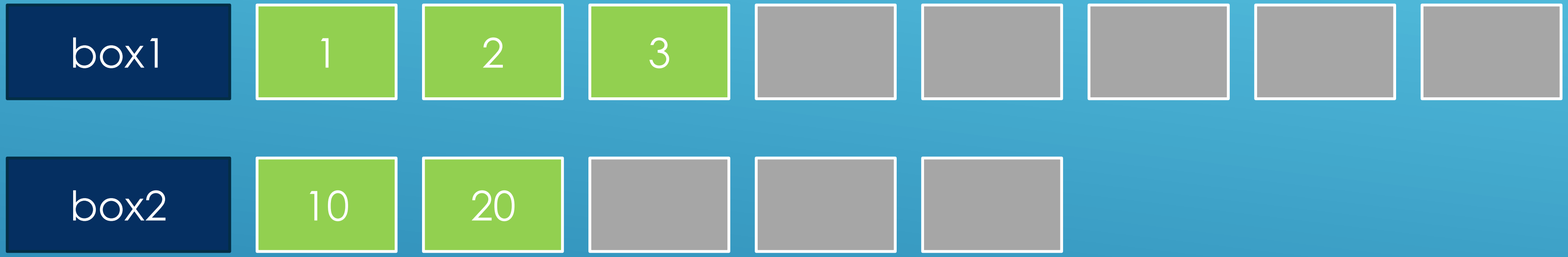
Removing elements



Operator+=

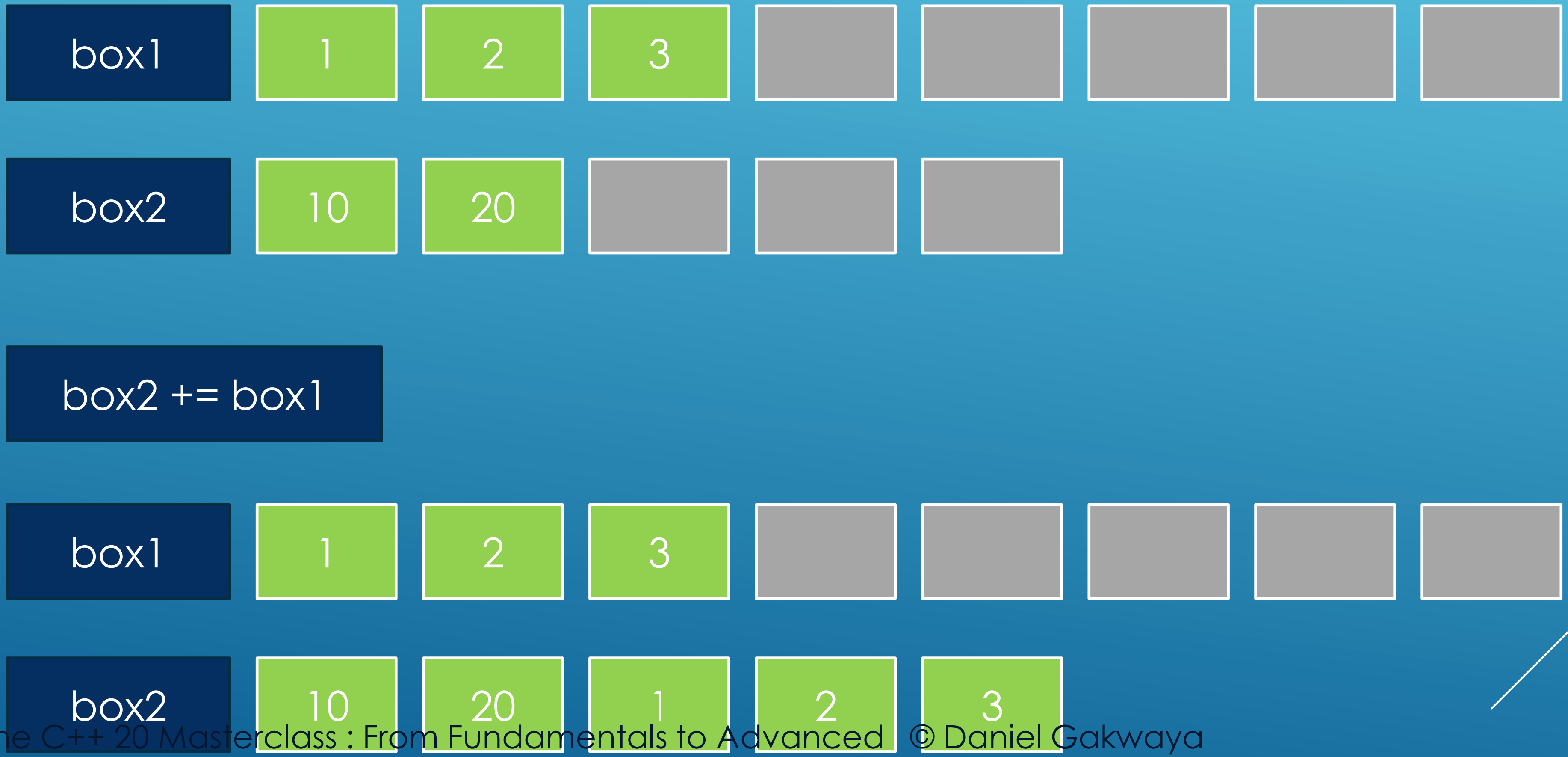


Operator+=

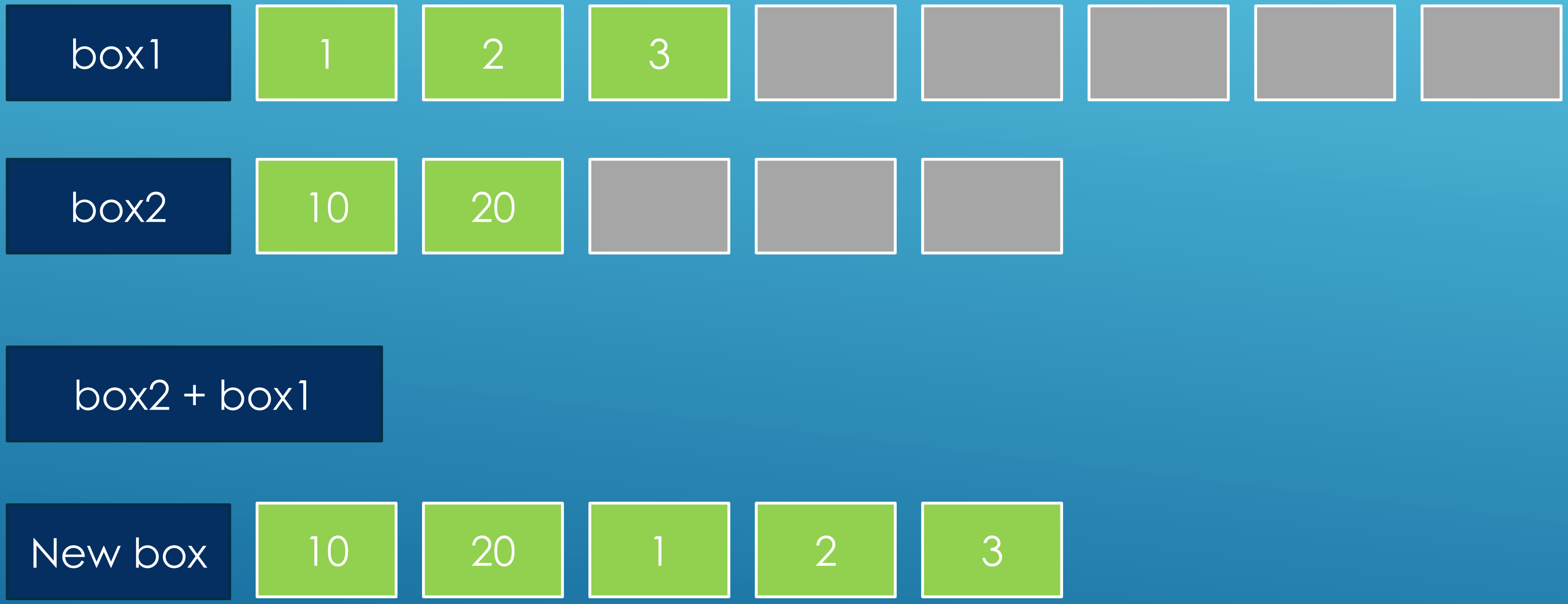


```
box2 += box1
```

Operator+=



Operator+



Class wrapping on top of raw array

```
class BoxContainer : public StreamInsertable
{
        typedef int value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 30;
public:
    BoxContainer(size_t capacity = DEFAULT_CAPACITY);
    BoxContainer(const BoxContainer& source);
    ~BoxContainer();

    //StreamInsertable Interface
    virtual void stream_insert(std::ostream& out)const;

    // Helper getter methods
    size_t size( ) const { return m_size; }
    size_t capacity() const{return m_capacity;};

    /* ...

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

Slide intentionally left empty

BoxContainer

Constructing, destructing

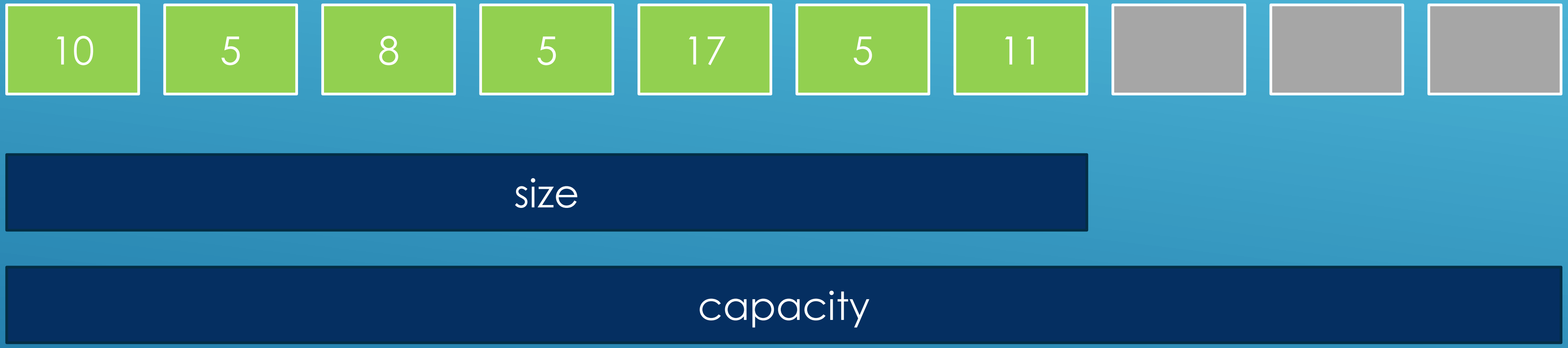
Building a box container

```
class BoxContainer : public StreamInsertable
{
        typedef int value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 30;
public:
    BoxContainer(size_t capacity = DEFAULT_CAPACITY);
    BoxContainer(const BoxContainer& source);
    ~BoxContainer();

    //StreamInsertable Interface
    virtual void stream_insert(std::ostream& out)const;

    // Helper getter methods
    size_t size( ) const { return m_size; }
    size_t capacity() const{return m_capacity;};

    /* ...

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

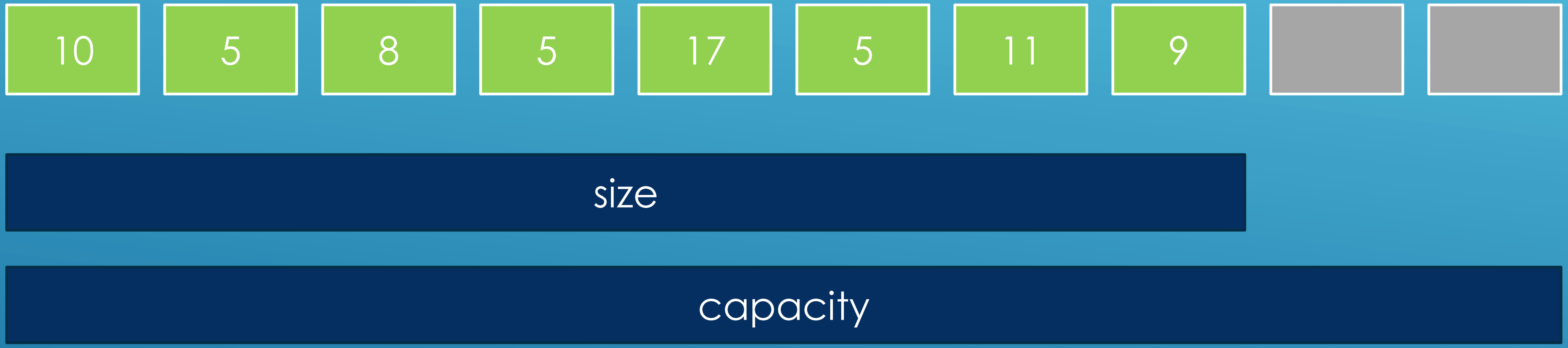
Slide intentionally left empty

BoxContainer

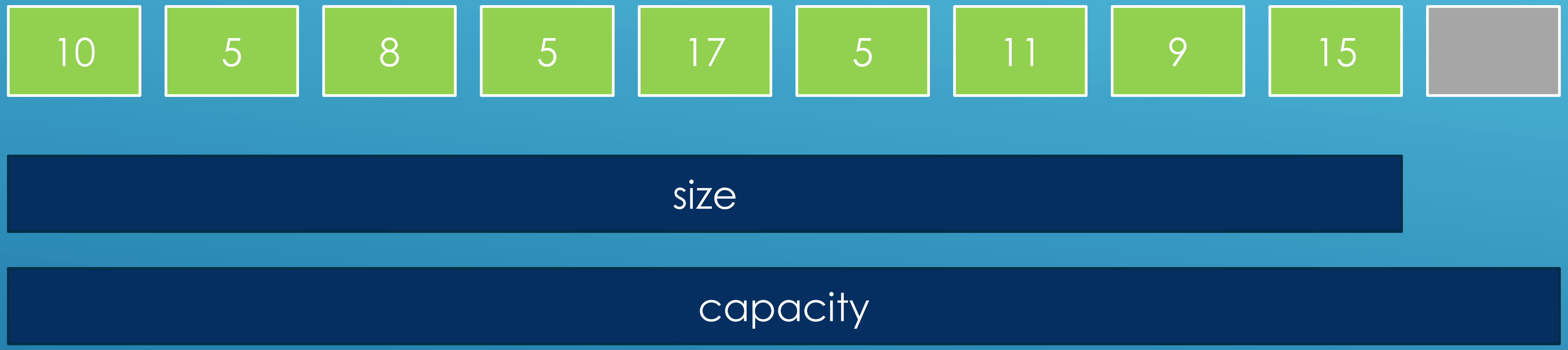
Adding Items


10
5
8
5
17
5
11
size
capacity

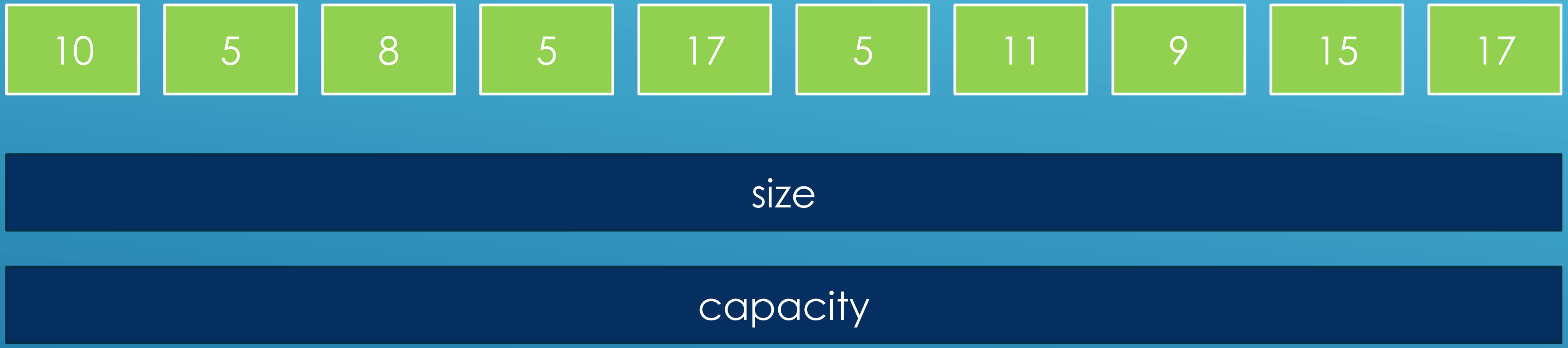
Adding elements



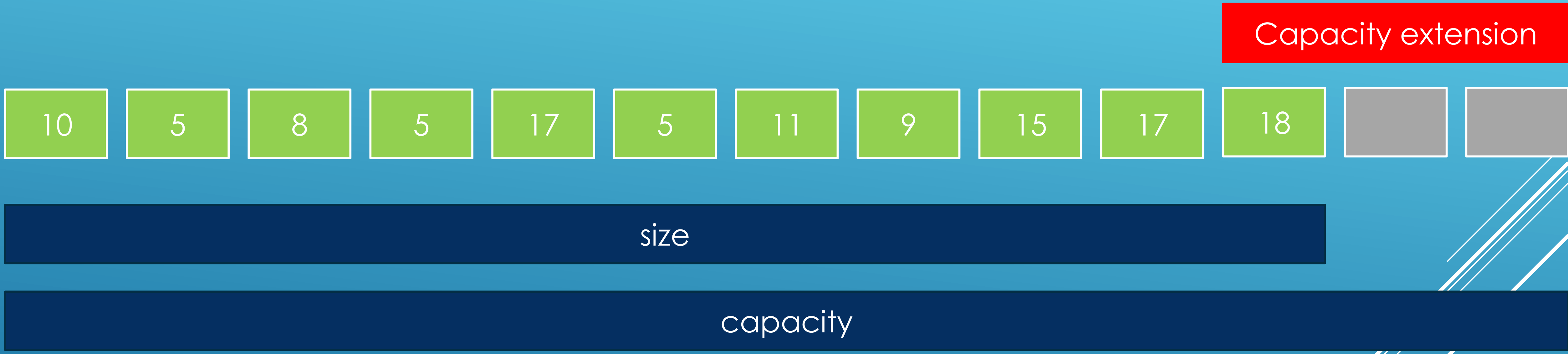
Adding elements



Adding elements



Adding elements



Expanding

```
void BoxContainer::expand(size_t new_capacity){
    std::cout << "Expanding to " << new_capacity << std::endl;
    value_type *new_items_container;

    if (new_capacity <= m_capacity)
        return; // The needed capacity is already there

    //Allocate new(larger) memory
    new_items_container = new value_type[new_capacity];

    //Copy the items over from old array to new
    for(size_t i{} ; i < m_size; ++i){
        new_items_container[i] = m_items[i];
    }

    //Release the old array
    delete [ ] m_items;

    //Make the current box wrap around the new array
    m_items = new_items_container;

    //Use the new capacity
    m_capacity = new_capacity;
}
```

Adding

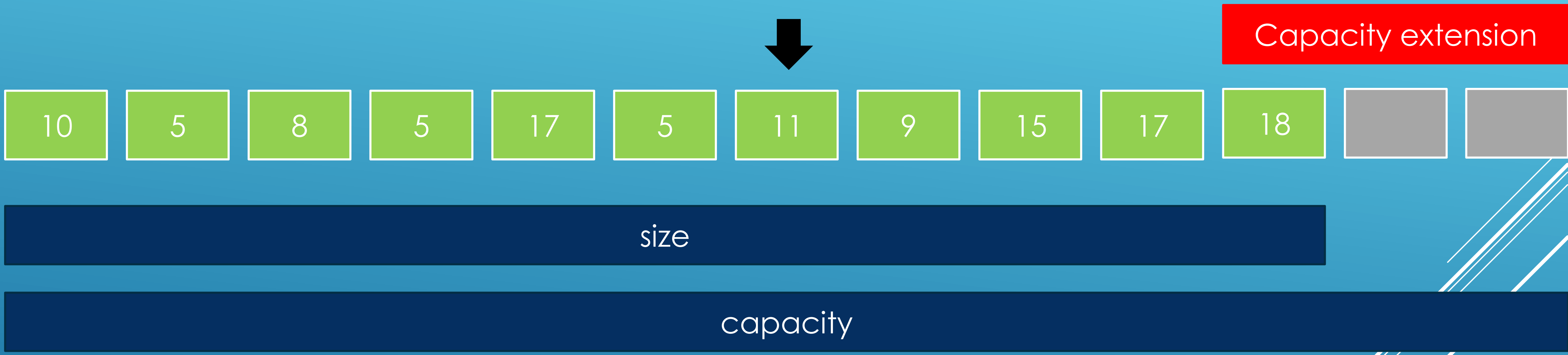
```
void BoxContainer::add(const value_type& item){
    if (m_size == m_capacity)
        expand(m_size+5); // Let's expand in increments of 5 to optimize on the calls to expand
    m_items[m_size] = item;
    ++m_size;
}
```

Slide intentionally left empty

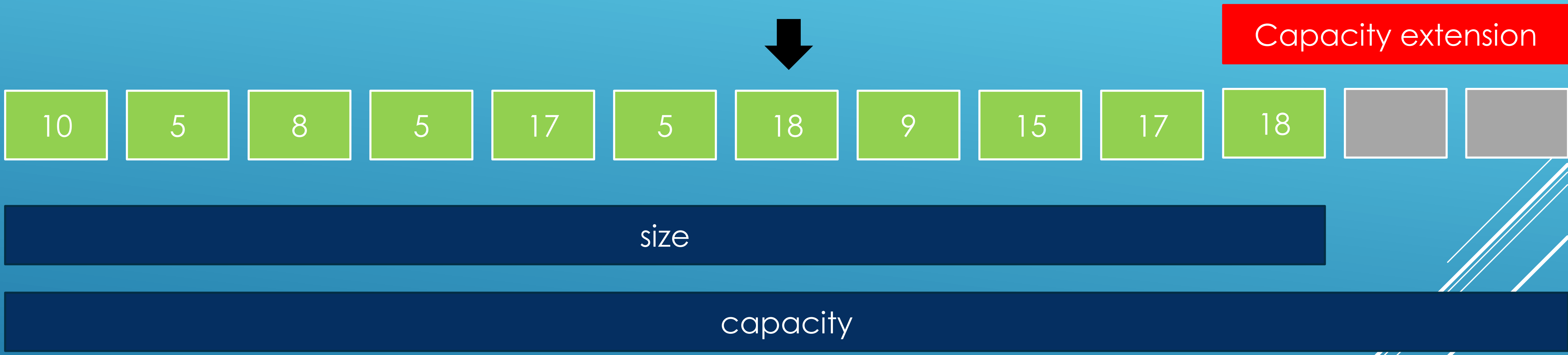
BoxContainer

Removing Items

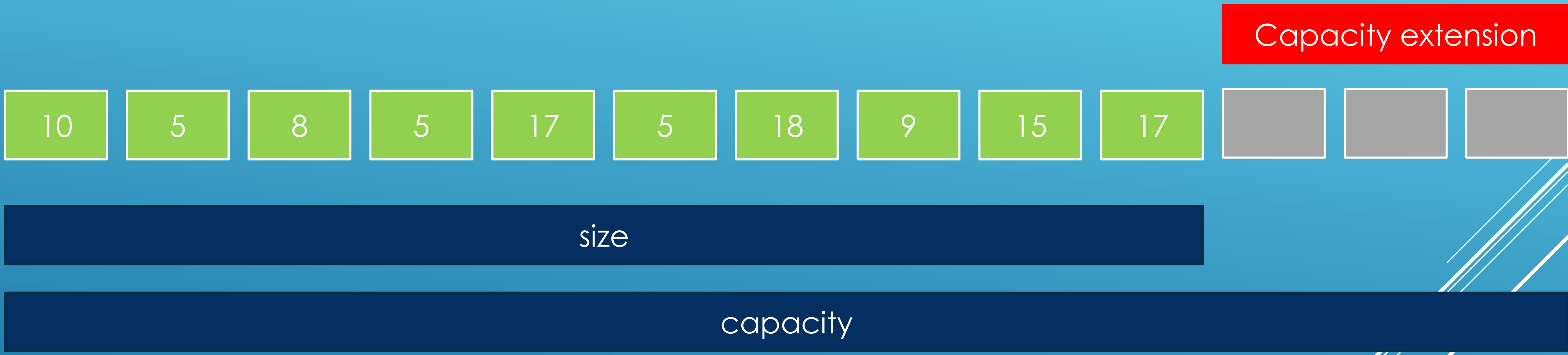
Removing elements



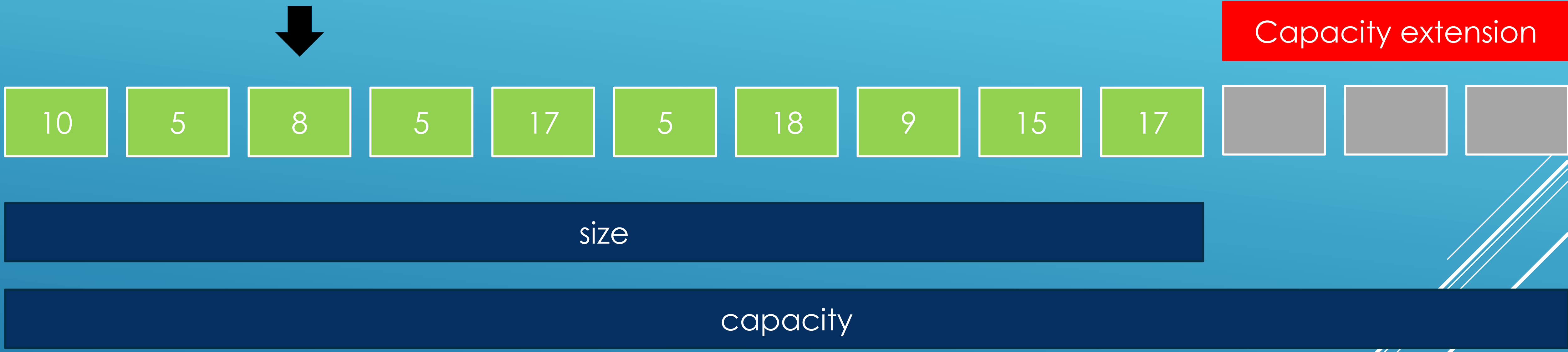
Removing elements

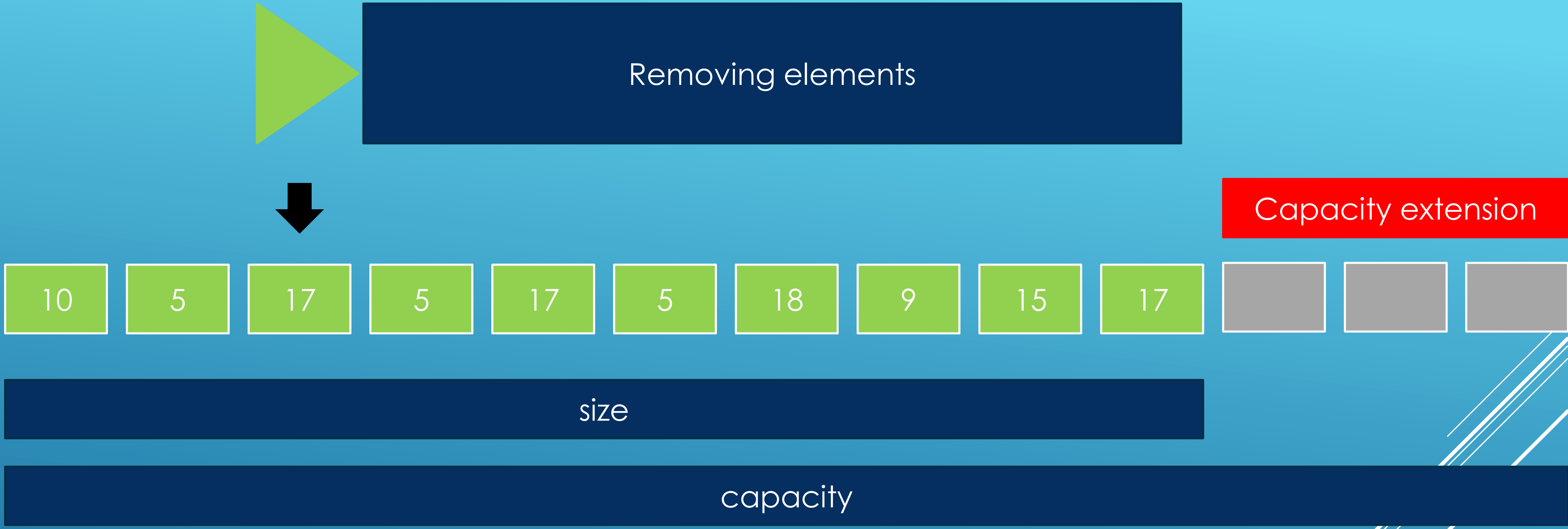


Removing elements

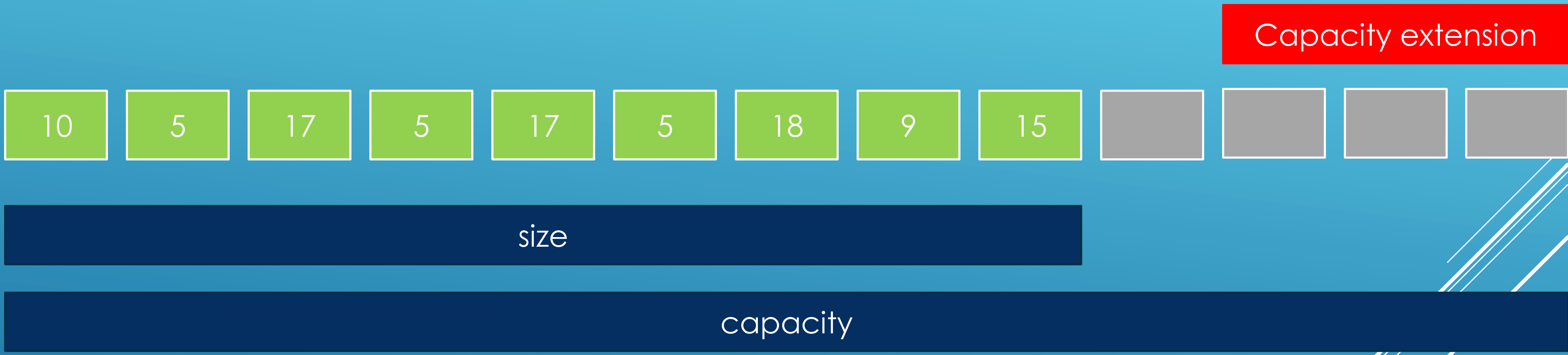


Removing elements

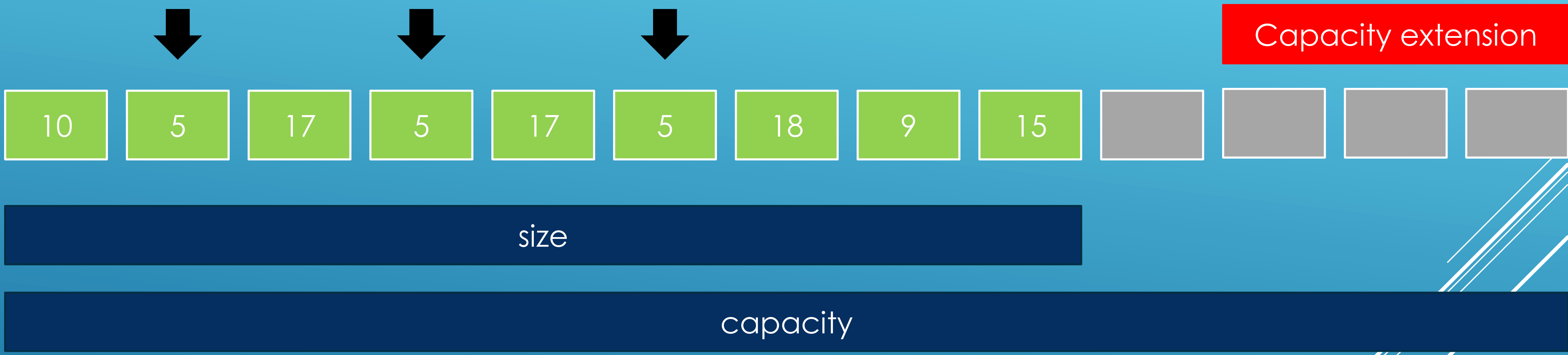



Removing elements
Capacity extension
10
5
17
5
17
5
18
9
15
17
size
capacity

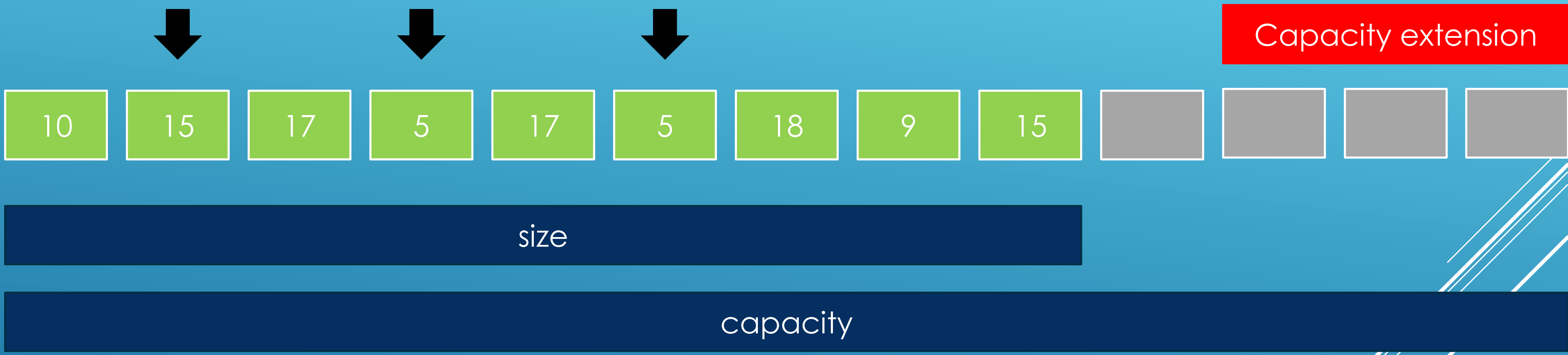
Removing elements



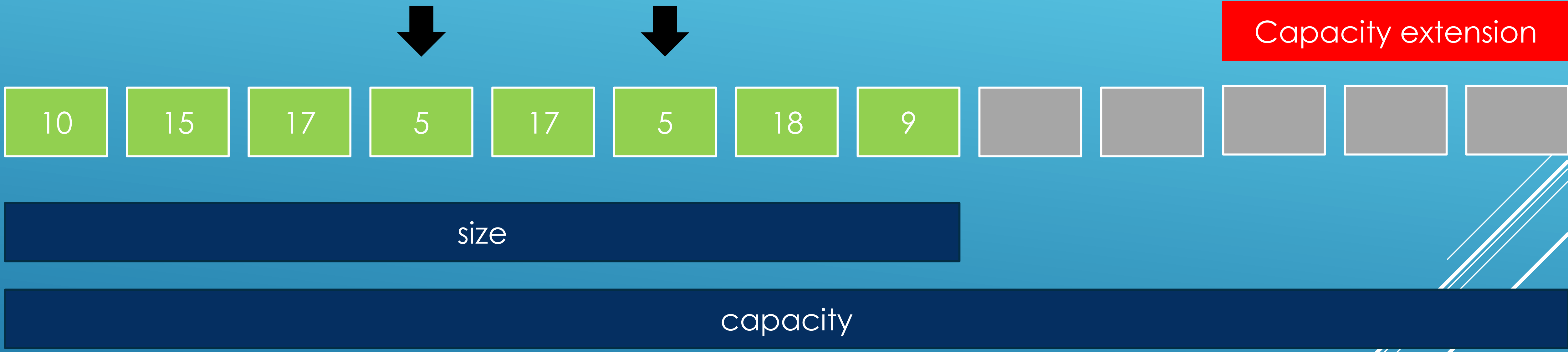
Removing elements



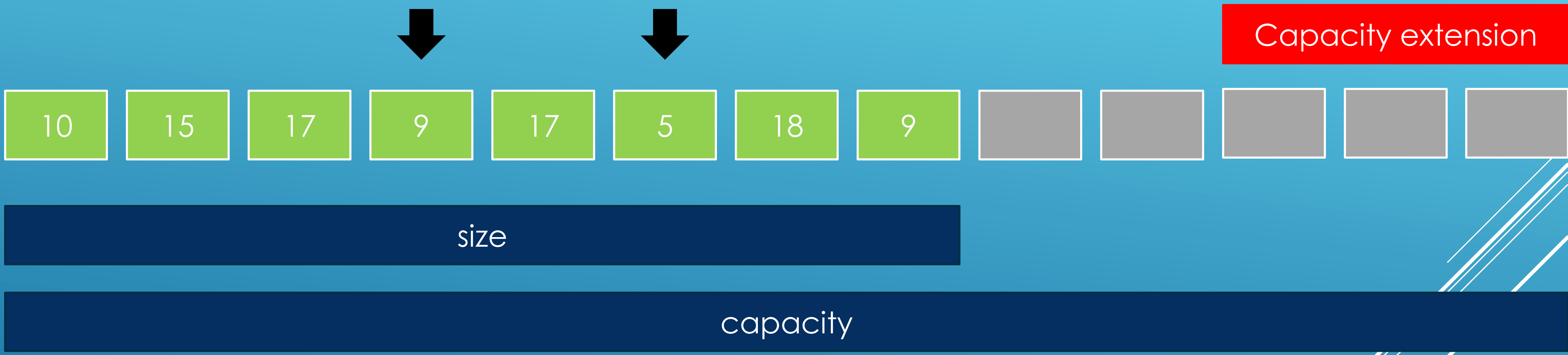
Removing elements



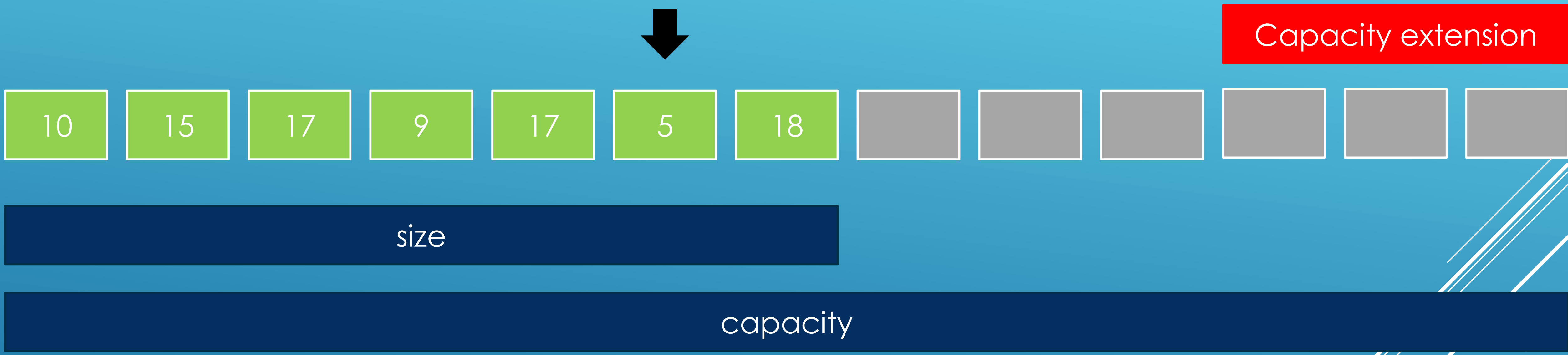
Removing elements



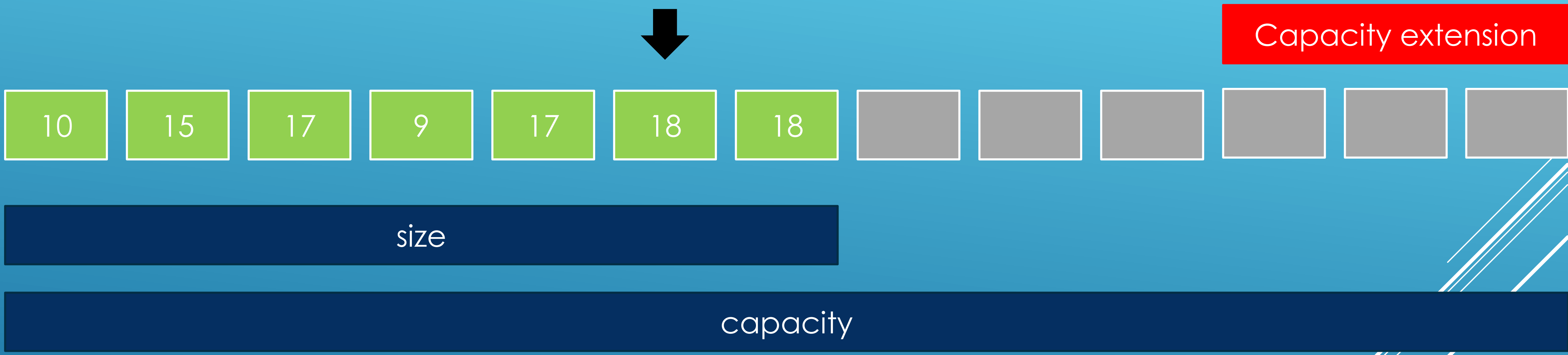
Removing elements



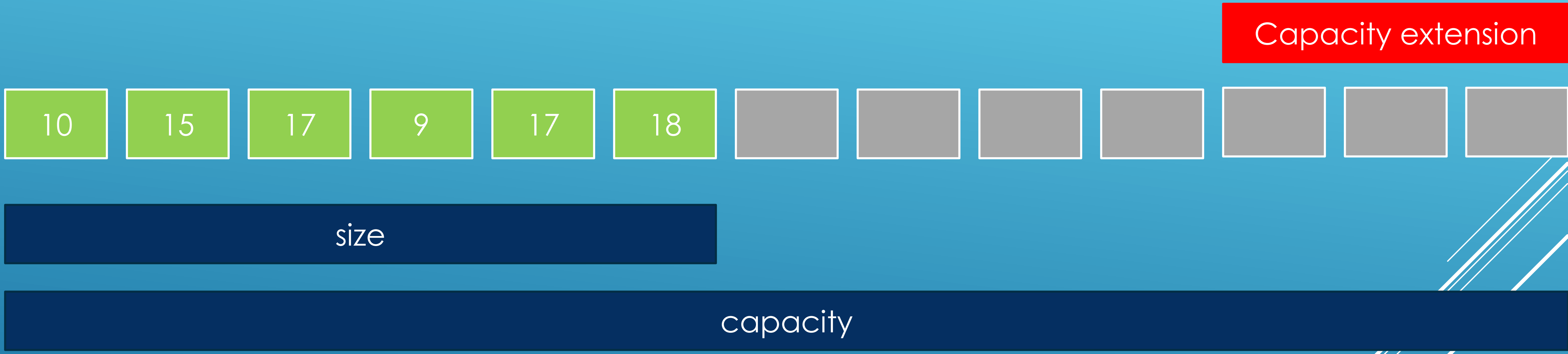
Removing elements



Removing elements

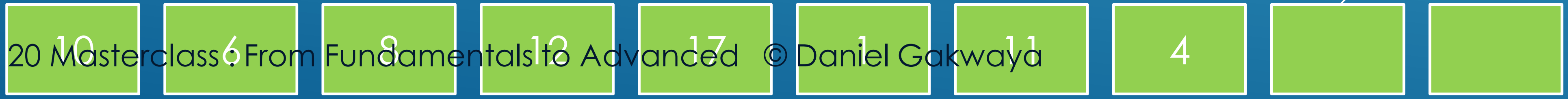


Removing elements

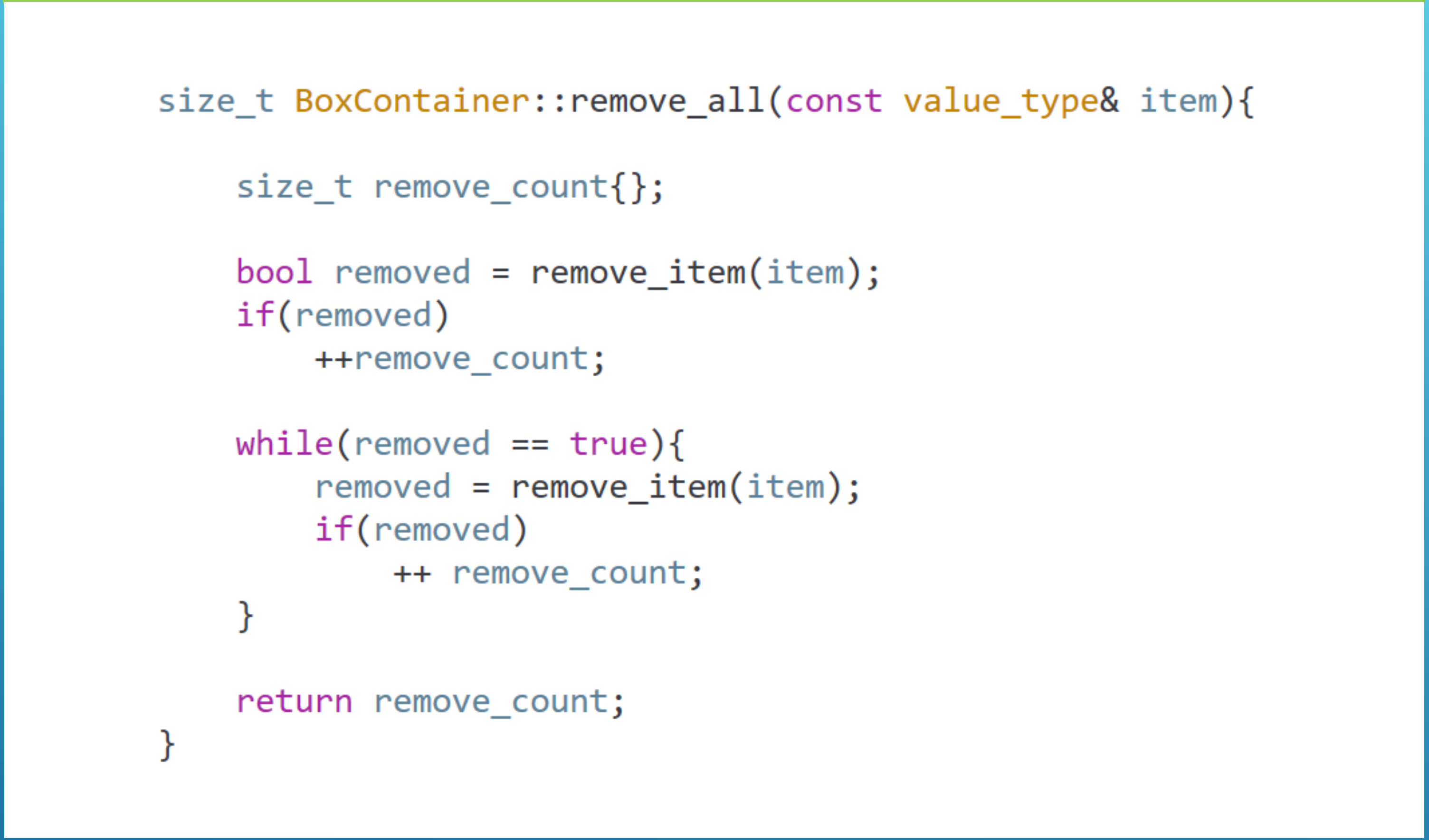


Removing one instance of an item

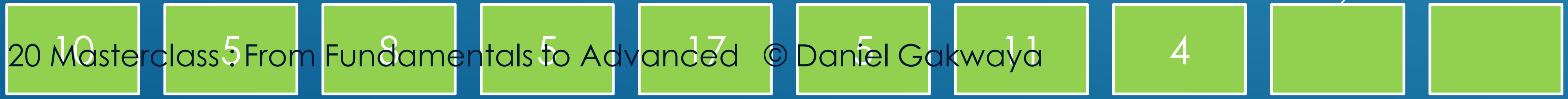
```
bool BoxContainer::remove_item(const value_type& item){

    //Find the target item
    size_t index {m_capacity + 999}; // A large value outside the range of the current
                                     // array
    for(size_t i{0}; i < m_size ; ++i){
        if (m_items[i] == item){
            index = i;
            break; // No need for the loop to go on
        }
    }

    if(index > m_size)
        return false; // Item not found in our box here

    //If we fall here, the item is located at m_items[index]

    //Overshadow item at index with last element and decrement m_size
    m_items[index] = m_items[m_size-1];
    m_size--;
    return true;
}
```



Removing all instances of an item : Multiple instances of same element



```
size_t BoxContainer::remove_all(const value_type& item){

    size_t remove_count{};

    bool removed = remove_item(item);
    if(removed)
        ++remove_count;

    while(removed == true){
        removed = remove_item(item);
        if(removed)
            ++ remove_count;
    }

    return remove_count;
}
```



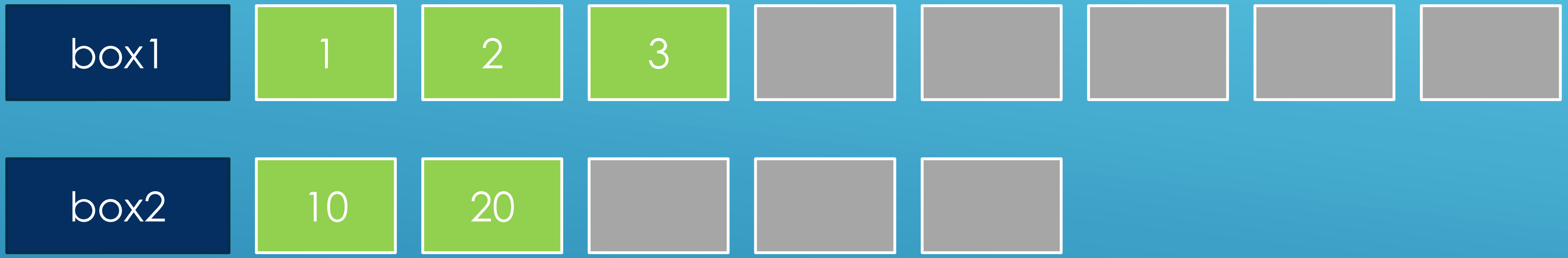
Slide intentionally left empty

BoxContainer

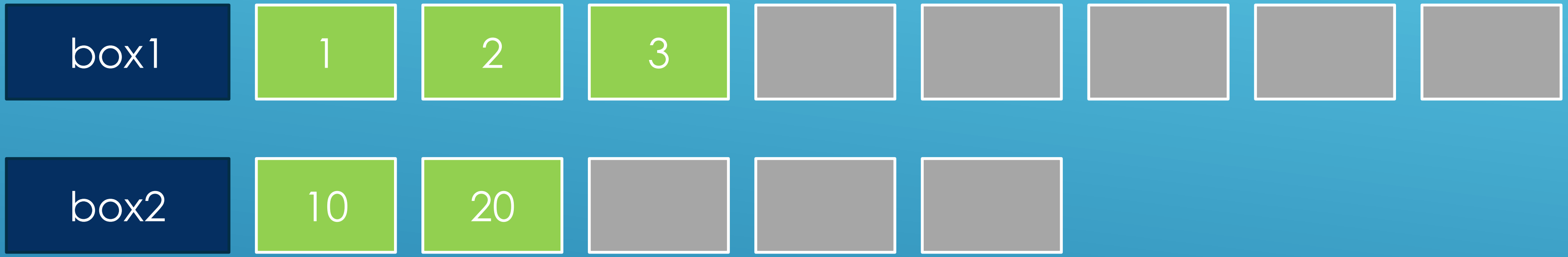
Other operators

- operator+=
- operator +
- operator=

Operator+=

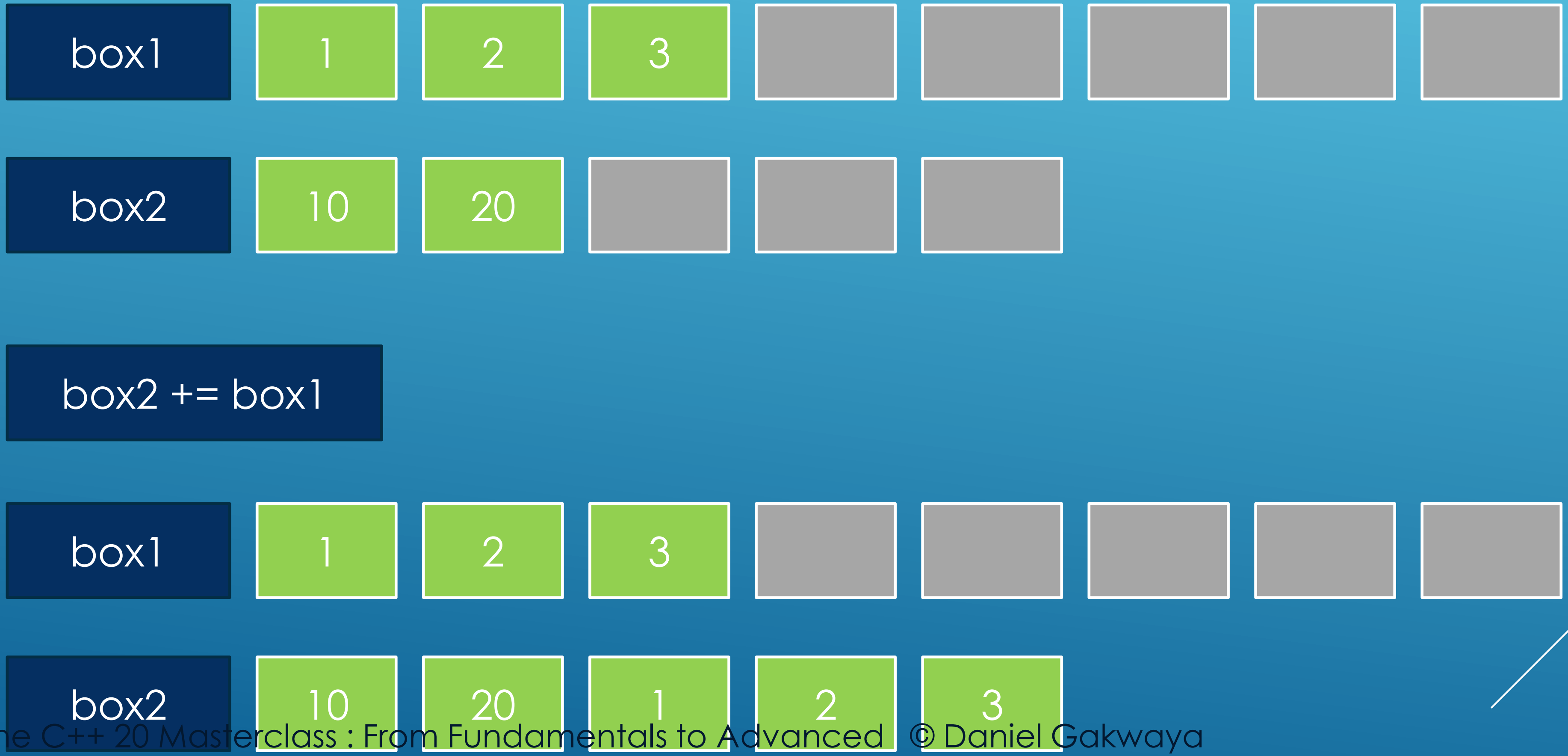


Operator+=

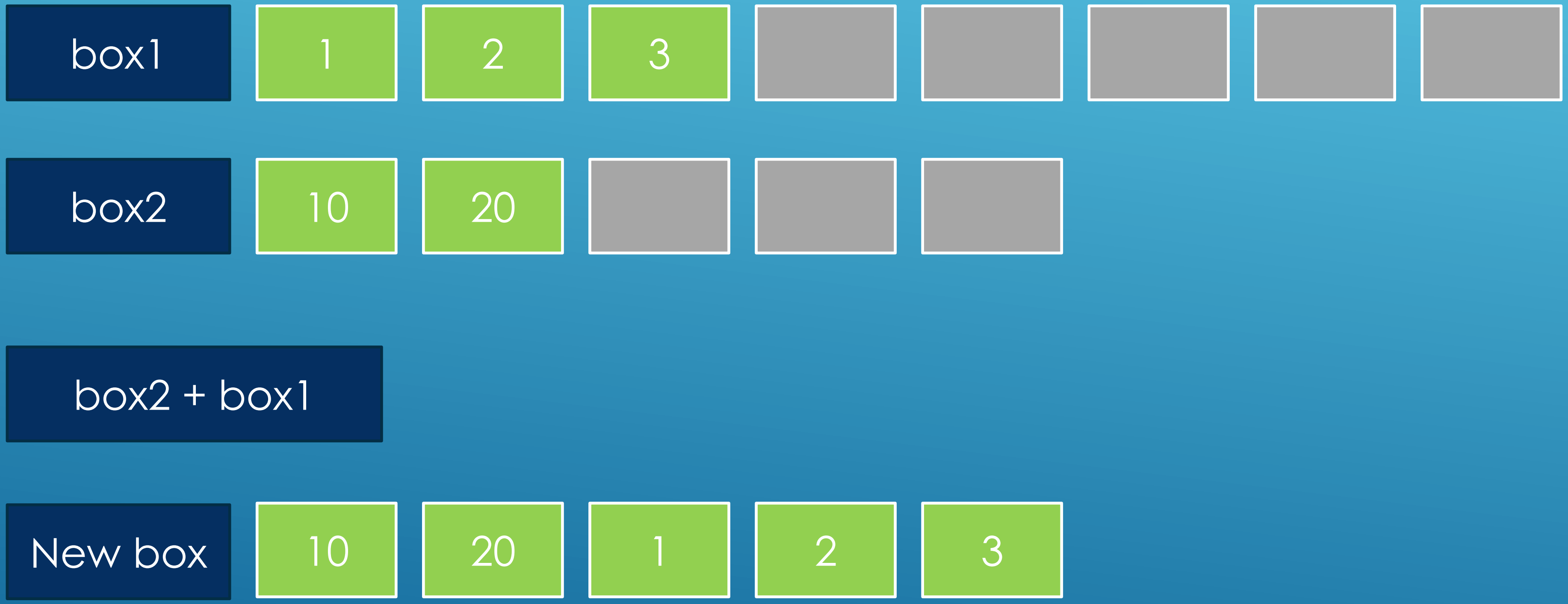


box2 += box1

Operator+=



Operator+



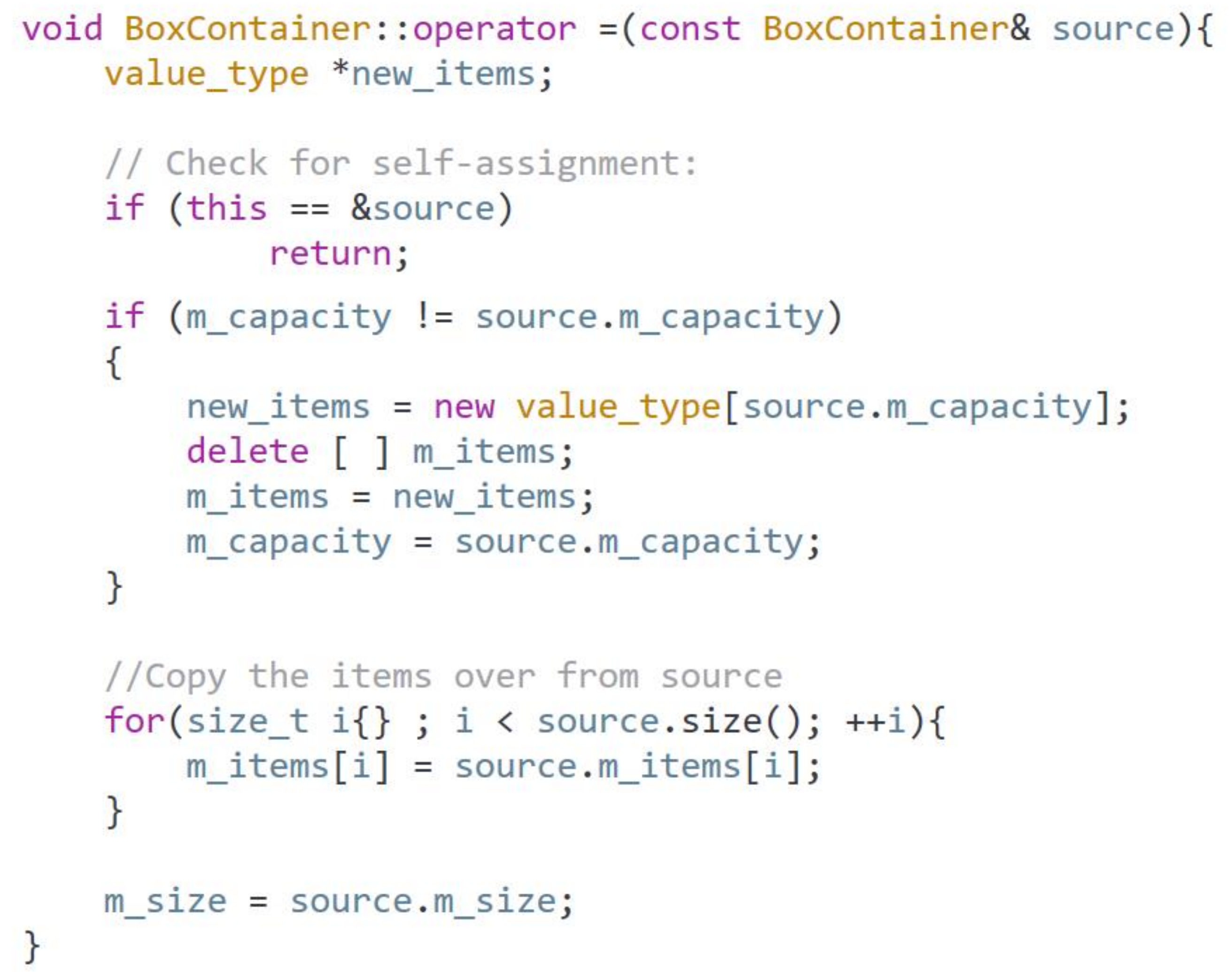
Operator+=

```
void BoxContainer::operator +=(const BoxContainer& operand){

    //Make sure the current box can acommodate for the added new elements
    if( (m_size + operand.size()) > m_capacity)
        expand(m_size + operand.size());

    //Copy over the elements
    for(size_t i{} ; i < operand.m_size; ++i){
        m_items [m_size + i] = operand.m_items[i];
    }

    m_size += operand.m_size;
}
```

Operator+

```
BoxContainer operator +(const BoxContainer& left, const BoxContainer& right){
    BoxContainer result(left.size( ) + right.size( ));
    result += left;
    result += right;
    return result;
}
```

Operator=



```
void BoxContainer::operator =(const BoxContainer& source){
    value_type *new_items;

    // Check for self-assignment:
    if (this == &source)
            return;

    if (m_capacity != source.m_capacity)
    {
        new_items = new value_type[source.m_capacity];
        delete [ ] m_items;
        m_items = new_items;
        m_capacity = source.m_capacity;
    }

    //Copy the items over from source
    for(size_t i{} ; i < source.size(); ++i){
        m_items[i] = source.m_items[i];
    }

    m_size = source.m_size;
}
```

Slide intentionally left empty

BoxContainer

Zooming out

```
. BoxContainer is a container class that provides the features
        . add_item
        . remove_item
        . remove_all

. We can add BoxContainer's up with :
        . +=
        . +

. Additionaly we can :
        . Stream insert BoxContainers with opreator<<
        . Copy construct BoxContainer's
        . Copy assign BoxContainer's
```

Slide intentionally left empty

BoxContainer

Storing in different types

IntContainer

```cpp
class IntContainer : public StreamInsertable
{
        typedef int value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    IntContainer(size_t capacity = DEFAULT_CAPACITY);
    IntContainer(const IntContainer& source);
    ~IntContainer();

    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

DoubleContainer

```
class DoubleContainer : public StreamInsertable
{
        typedef double value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    DoubleContainer(size_t capacity = DEFAULT_CAPACITY);
    DoubleContainer(const DoubleContainer& source);
    ~DoubleContainer();
    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

CharContainer

```
class CharContainer : public StreamInsertable
{
        typedef char value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    CharContainer(size_t capacity = DEFAULT_CAPACITY);
    CharContainer(const CharContainer& source);
    ~CharContainer();
    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

Slide intentionally left empty

BoxContainer

```
. BoxContainer is a container class that provides the features
        . add_item
        . remove_item
        . remove_all

. We can add BoxContainer's up with :
        . +=
        . +

. Additionaly we can :
        . Stream insert BoxContainers with opreator<<
        . Copy construct BoxContainer's
        . Copy assign BoxContainer's
```

Class wrapping on top of raw array

```
class BoxContainer : public StreamInsertable
{
        typedef int value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 30;
public:
    BoxContainer(size_t capacity = DEFAULT_CAPACITY);
    BoxContainer(const BoxContainer& source);
    ~BoxContainer();

    //StreamInsertable Interface
    virtual void stream_insert(std::ostream& out)const;

    // Helper getter methods
    size_t size( ) const { return m_size; }
    size_t capacity() const{return m_capacity;};

    /* ...

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

IntContainer

```
class IntContainer : public StreamInsertable
{
        typedef int value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    IntContainer(size_t capacity = DEFAULT_CAPACITY);
    IntContainer(const IntContainer& source);
    ~IntContainer();

    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

DoubleContainer

```
class DoubleContainer : public StreamInsertable
{
        typedef double value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    DoubleContainer(size_t capacity = DEFAULT_CAPACITY);
    DoubleContainer(const DoubleContainer& source);
    ~DoubleContainer();
    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

CharContainer

```
class CharContainer : public StreamInsertable
{
        typedef char value_type; // Allows us to change what's stored in the vector on the fly
                                // Can make it store int, double,...
        static const size_t DEFAULT_CAPACITY = 5;
        static const size_t EXPAND_STEPS = 5;
public:
    CharContainer(size_t capacity = DEFAULT_CAPACITY);
    CharContainer(const CharContainer& source);
    ~CharContainer();
    /* ...
private :
    void expand(size_t new_capacity);

private :
    value_type * m_items;
    size_t m_capacity;
    size_t m_size;
};
```

Class templates to the rescue

Slide intentionally left empty