**Описание контейнера**.

Документ содержит описание контейнера Int\_Array, в который могут помещаться объекты класса int. Число объектов, размещенных в контейере, может изменяться, такое число будем называть текущим размером контейнера. Пустой контейнер имеет размер 0. Объекты, размещенные в контейнере, имеют номера от 0 до n – 1, где n – текущий размер контейнера.

Описание контейнера на C++ содержится в файле int\_array.h.

**Конструкторы контейнера**.

Int\_Array();

Создает пустой контейнер.

Int\_Array(unsigned number, int value);

Создает контейнер из number элементов, имеющих однинаковое значение value.

Int\_Array(unsigned number, int\* a);

Создает контейнер размером number, помещает в контейнер number элементов масива a.

**Примечание**. Определен также конструктор копирования.

**Основные методы и операторы**.

При выполнении методов, описанных в этом разделе и имеющих параметр index, значение index должно быть меньше текущего размера контейнера, в противном случае выбрасывается исключение Bad\_Index. В частности, это происходит при применении таких методов к пустому контейнеру.

unsigned size();

Метод возвращает текущий размер контейнера. Если контейнер пуст, возвращает 0. Не выбрасывает исключений.

Определен оператор [], возвращающий ссылку на элемент контейнера с заданным номером. Номер должен быть меньше размера контейнера, в противном случае выбрасывается исключение Bad\_Index.

int& operator [] (unsigned index);

Определен оператор присваивания.

Int\_Array& operator = (const Int\_Array& source);

Для контейнера определен бинарный оператор <<, помещающий в поток последовательность представлений значений элементов, разделенных пробелами.

ostream& operator << (ostream& output, const Int\_Array& a);

**Методы и операторы, расширяющие контейнер**.

При выполнении методов, описанных в этом разделе и имеющих параметр index, значение index не должно превышать текущего размера контейнера, в противном случае выбрасывается исключение Bad\_Index.

void insert(unsigned index, int value);

Помещает новый элемент со значением value в позицию с номером index. При этом размер контейнера увеличивается на 1, номера всех элементов, следующих за элементом с номером index (если такие элементы имеются), также увеличиваются на 1. Если контейнер c пуст, то insert(0, value) помещает элемент в контейнер, если контейнер c непуст, то c.insert(0, value) помещает элемент в начало, c.insert(c.size(), value) помещает элемент в конец.

void replicate(unsigned index, unsigned number, int value);

Вставляет в контейнер, начиная с элемента с номером index, number значений value. Размер контейнера увеличивается на number, номера всех элементов, следующих за элементом с номером index (если такие элементы имеются), также увеличиваются на number. Если контейнер c пуст, то помещает в него number элементов со значением value, если контейнер c непуст, то выполнение c.replicate(0, number, value) помещает number значений value в начало контейнера, выполнение c.replicate(c.size(), number, value) помещает number значений value в конец контейнера.

void insert(unsigned index, Int\_Array& array);

Вставляет в контейнер, начиная с элемента с номером index, все элементы контейнера array. Размер контейнера увеличивается на размер контейнера array, номера всех элементов, следующих за элементом с номером index (если такие элементы имеются), также увеличиваются на размер контейнера array. Если контейнер c пуст, помещает в него элементы контейнера array. В частности, c.include(0, array) выполняет конкатенацию контейнера array с данным контейнером, c.include(c.size(), array) выполняет конкатенацию данного контейнера с контейнером array.

Определен также оператор конкатенации, его выполнение для контейнера с эквивалентно выполнению c.insert(c.size(), source).

Int\_Array& operator += (const Int\_Array& source);

**Методы, сужающие контейнер**.

При выполнении методов, описанных в этом разделе и имеющих параметр index, значение index должно быть меньше текущего размера контейнера, в противном случае выбрасывается исключение Bad\_Index. В частности, это происходит при применении таких методов к пустому контейнеру.

int remove(unsigned index);

Удаляет элемент контейнера с номером index, возвращает его значение. При этом размер контейнера уменьшается на 1, номера всех элементов, следюущих за элементом с номером index (если такие элементы имеются), также уменьшаются на 1. Если контейнер c содержит один элемент, то c.remove(0) возвращает значение элемента, контейнер становится пустым, для любого конейнера c с.remove(0) удаляет начачальный элемент, с.remove(с.size() - 1) удаляет последний элемент.

void remove(unsigned index, unsigned number);

Удаляет из контейнера, начиная с номера index, number элементов. Если значение index + number равно или превышает текущий размер контейнера, то удаляет все элементы, начиная с номера index до конца контейнера. Если number == 0, то никаких действий не предпринимается.

Cледующий метод удаляет все элементы, начиная c номера index до конца, его выполнение которого для контейнера c эквивалентно c.remove(index, c.size() – index).

void remove\_from(unsigned index);

Следующий метод очищает контейнер, не выбрасывает исключений. Метод применим у пустому контейнеру.

void clear();

**Методы обработки контейнера**.

Методы, описанные в этом разделе, позволяют выполнять с элементами контейнера некоторые операции, для выполнения таких операций может использоваться вспомогательный объект, который должен быть экземпляром класса – наследника класса Int\_Array\_Handler.

Класс Int\_Array\_Handler определен следующим образом.

class Int\_Array\_Handler

{

public:

void handle(Int\_Array& a, unsigned n);

bool condition(Int\_Array& a unsigned n);

};

Метод handle не предпринимает никаких действий, метод condition возвращает false. Методы могут быть переопределены наследниками класcа In\_Array\_Handler.

При выполнении методов, описанных в этом разделе и имеющих параметр index, значение index не должно превышать текущего размера контейнера, в противном случае выбрасывается исключение Bad\_Index. При применении у к пустому контейнеру с index = 0 никаких действий не предпринимается.

unsigned apply( unsigned index, Int\_Array\_Handler& handler);

Описанный выше метод с элементами контейнера с номерами n, начиная с номера index до конца контейнера, выполняе следующее: если handler.condition(\*this, n) возвращает true, то выполнение метода завершается, метод возвращает n, в противном случае выполняется handler.handle(\*this, n). Если ни для какого номера n handler.condition(\*this, n) не возвращает true, то метод возвращает размер контейнера.

Следующий метод возвращает apply(0, handler), не выбрасывает исключений, если их не выбрасывает обработчик.

void apply(Int\_Array\_Handler& handler);

Следующий метод выполняет поиск элемента со значением value, начиная с номера index. Если значение найдено, возвращает true, \*n имеет значение индекса найденного элемента, в противном случае возвращает false, \*n имеет значение размера контейнера. Не выбрасывает исключений.

bool found(int value, unsigned index, unsigned\* n);

Следующий метод выполняет поиск элемента со значением value c начала контейнера. Возвращает found(value, 0, n). Если контейнер пуст, всегда возвращает false. Не выбрасывает исключений.

bool found(int value, unsigned\* n);

**Исключения**.

В случае недопустимого значения индекса выбрасывается исключение Bad\_Index, класс Bad\_Index определен следующим образом.

class Bad\_Index : public exception

{

public:

Bad\_Index(unsigned index);

unsigned value();

virtual const char\* what() const noexcept override;

private:

unsigned bad\_index;

const char\* text = nullptr;

};

Метод value возвращает значение интекса. Метод what возвращает строку в следующем формате.

error: invaid Int\_Array index <значение индекса>.