Лабораторная работа №20 «Контейнеры»

Задание: Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции: [] – доступа по индексу; int() – определение размера вектора; \* вектор – умножение элементов векторов a[i]\*b[i]; + n – переход вправо к элементу с номером n ( с помощью класса-итератора).

Цель работы: написать программу, которая реализует указанные в варианте операции.

#include <iostream>

#include <ctime>

#include <vector>

#include <stdexcept>

class Vector

{ // Класс Veector

private:

int\* array;

int size;

public:

Vector(int size = 2 + rand() % 9)

{// Констркутор, который определяет и инициализирует динамический массив

array = new int[size];

this->size = size;

for(int i = 0; i < size; i++)

array[i] = rand() % 10;

}

~Vector()

{// Деструктор

delete[] array;

}

const int Get\_size()

{// Функция возвращающая размер массива

return size;

}

const int& operator [] (int &index)

{// Перегрузка оператора []

return array[index];

}

const void print\_array()

{// Функция для вывода массива на экран

for (int i = 0; i < size; i++)

std::cout << array[i] << ' ';

}

};

class RightIterator

{// Класс Итератор

private:

std::vector<int>::iterator current;

std::vector<int>::iterator end;

size\_t index;

public:

RightIterator(std::vector<int>::iterator begin, std::vector<int>::iterator end, size\_t n)

: current(begin), end(end), index(n)

{// Конструктор с параметрами, который инициализирует поля класса и производит итерацию на указанный индекс

if (index >= static\_cast<size\_t>(std::distance(begin, end))) // Проверка на диапозон

throw std::out\_of\_range("Index out of range");

std::advance(current, n); // Изменяет положение итератора на указаный индекс

}

int next()

{// Функция, которая возвращает значение индекса итератора, иначе обрабатывает исключение(71 стр)

if (current != end)

{

int value = \*current;

++current;

return value;

}

else

throw std::out\_of\_range("Iterator reached the end");

}

};

int main()

{

srand(time(NULL));

Vector vector;

std::cout << "Full array: ";

vector.print\_array();

int first\_index = rand() % (vector.Get\_size());

int second\_index = rand() % (vector.Get\_size());

std::cout << "\nfirst\_index: " << first\_index << ' ' << "second\_index: " << second\_index << std::endl;

std::cout << std::endl << "a[first\_index] \* b[second\_index] = " << vector[first\_index] \* vector[second\_index] << std::endl;

std::cout << std::endl;

std::vector<int> array(vector.Get\_size());

for (int i = 0; i < vector.Get\_size(); i++)

{

array[i] = vector[i];

}

int n;

std::cout << std::endl << "Enter +n: ";

std::cin >> n;

try

{

RightIterator it(array.begin(), array.end(), n);

std::cout << "Current element: " << array[0] << ' ' << "Values to the right of element at index " << n << ": " << it.next() << std::endl;

}

catch (const std::out\_of\_range& e)

{

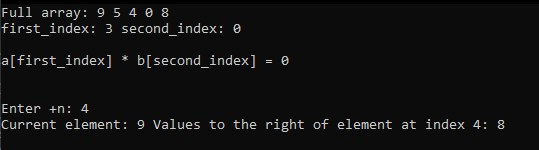
std::cout << e.what() << std::endl;

}

return 0;

}

Рисунок 1. Результат выполнения



Постановка задачи:

1.Определить класс-контейнер.

class Vector

{ // Класс Veector

private:

int\* array;

int size;

public:

Vector(int size = 2 + rand() % 9)

{// Констркутор, который определяет и инициализирует динамический массив

array = new int[size];

this->size = size;

for(int i = 0; i < size; i++)

array[i] = rand() % 10;

}

~Vector()

{// Деструктор

delete[] array;

}

const int Get\_size()

{// Функция возвращающая размер массива

return size;

}

const int& operator [] (int &index)

{// Перегрузка оператора []

return array[index];

}

const void print\_array()

{// Функция для вывода массива на экран

for (int i = 0; i < size; i++)

std::cout << array[i] << ' ';

}

};

2.Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

Vector(int size = 2 + rand() % 9)

{// Констркутор, который определяет и инициализирует динамический массив

array = new int[size];

this->size = size;

for(int i = 0; i < size; i++)

array[i] = rand() % 10;

}

Это конструктор, который принимает параметр по умолчанию, если в него не передаются данные, он создает динамический массив, затем передает размер массива в поле класса и инициализирует массив с помощью ДСЧ.

~Vector()

{// Деструктор

delete[] array;

}

Это деструктор, который очищает массив м класс соответственно.

3.Перегрузить операции, указанные в варианте.

const int& operator [] (int &index)

{// Перегрузка оператора []

return array[index];

}

Эта функция перегружает оператор [] для получения значения указанного под передаваемым индексом.

Это единственный оператор, который является перегруженным в данной версии программы.

4.Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.

class RightIterator

{// Класс Итератор

private:

std::vector<int>::iterator current;

std::vector<int>::iterator end;

size\_t index;

public:

RightIterator(std::vector<int>::iterator begin, std::vector<int>::iterator end, size\_t n)

: current(begin), end(end), index(n)

{// Конструктор с параметрами, который инициализирует поля класса и производит итерацию на указанный индекс

if (index >= static\_cast<size\_t>(std::distance(begin, end))) // Проверка на диапозон

throw std::out\_of\_range("Index out of range");

std::advance(current, n); // Изменяет положение итератора на указаный индекс

}

int next()

{// Функция, которая возвращает значение индекса итератора, иначе обрабатывает исключение(71 стр)

if (current != end)

{

int value = \*current;

++current;

return value;

}

else

throw std::out\_of\_range("Iterator reached the end");

}

};

Это класс итератор, который имеет 3 поля current, end и index. Этот класс нужен для того, чтобы обработать класс Vector и вернуть значение под передаваемым индексом, если оно существует, а иначе вызвать исключение.

5.Написать тестирующую программу, иллюстрирующую выполнение операций.

Текст указан выше.

Содержание отчета

1. Постановка задачи (общая и для конкретного варианта);

Описано в постановка задачи.

2. Анализ задачи:

• Определения функций для реализации поставленных задач;

Описано выше.

• Определение функции main();

Создается объект класса Vector под именем vector, далее демонстрируется функция a[i]\*b[i] и в ней же демонстрируется [], затем демонстрируется +n с помощь класса итератора.

• Описание класса-контейнера.

Описано выше.

• Определение компонентных функций;

Не используется.

Вывод по работе:

В ходе выполнения работы были получены навыки по созданию классов, а также знания о спецификаторах доступа, полях и методов, селекторах и модификаторах.