МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

Подразделение кафедра информационной безопасности

(наименование подразделения)

ОТЧЕТ ПО ПРАКТИКЕ

Вид (тип) практики	учебная (ознакомительная)
Курс	первый
Направление подготовки (специальность)	информационная безопасность АС
Ф.И.О. обучающегося	Павлова Виктория Сергеевна
Место прохождения практики	ТулГУ, кафедра Информационной безопасности
Период прохождения практики	с 07 июля 2022 г. по 20 июля 2022 г.

Руководитель практической подготовки
(руководитель практики) от профильной организации

_____ _____

(Ф.И.О., должность) *(подпись)*

М. П.

Руководитель практической подготовки
(руководитель практики) от университета

_____ _____

(Ф.И.О., должность) *(подпись)*

Тула 20 22 г.

МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «Тульский государственный университет»

просп. Ленина, 92, г. Тула, 300012 тел. (4872)25-70-89

УЧЕТНАЯ КАРТОЧКА

прохождения учебной (ознакомительной) практики

(вид и тип практики)

ФИО обучающегося: Павлова Виктория Сергеевна
Направление подготовки (специальность): Информационная безопасность автоматизированных систем
Курс: первый Номер группы: 230711
Место проведения практики: г. Тула, ТулГУ, кафедра ИБ

(город, организация)

Индивидуальное задание:
Структура данных unordered set

СВЕДЕНИЯ О ПРАКТИКЕ

Дата прибытия на практику
«07» июля 2022 г. _____
(должность и подпись уполномоченного лица) М.П.

В период прохождения практики зачислен на вакантную должность

«____» _____ 20___ г. _____
(должность и подпись уполномоченного лица) М.П.

Дата окончания практики
«20» июля 2022 г. _____
(должность и подпись уполномоченного лица) М.П.

Раздел руководителя практической подготовки (руководителя практики) от профильной организации

(заполняется при прохождении практики в профильной организации)

Отзыв о прохождении практики

_____

_____

_____

_____

_____

Руководитель практической подготовки (руководитель практики) от профильной организации_____

(должность)

_____ _____

(Ф.И.О.) *(подпись)*

М.П.

Раздел руководителя практической подготовки (руководителя практики) от университета

Отзыв о прохождении практики

_____

_____

_____

_____

_____

Результат промежуточной аттестации обучающегося по практике

Руководитель практической подготовки (руководитель практики) от университета_____

(должность)

_____ _____

(Ф.И.О.) *(подпись)*

Оглавление

Введение

В основу любой современной программы заложен алгоритм, который оперирует разнообразными структурами данных. В зависимости от конкретных потребностей и задач вводимые данные необходимо преобразовывать к тому или иному виду, поскольку они не всегда сразу поддаются обработке. Для упрощения этого процесса используются встроенные или самописные шаблоны и классы, позволяющие автоматизировать и упростить процесс создания программ.

Стандартная библиотека шаблонов языка программирования C++ называется STL и предоставляет различные варианты реализации таких структур данных, как контейнеры, необходимых для хранения коллекций связанных объектов. Контейнеры являются шаблонами классов, и у них есть некоторые особенности, например, итерация элементов в контейнере и доступ к отдельным элементам осуществляются с помощью итераторов. [1]

В качестве темы для изучения в рамках данной ознакомительной практики был выбран один из шаблонных контейнеров STL под названием unordered_set. Подробнее об особенностях его реализации и возможностях практического применения изложено в разделах с практической и теоретической частью.

В качестве практической части в рамках отчёта об итогах ознакомительной практики приведены самостоятельно разработанные и решённые задачи с использованием изученной структуры данных. В качестве теоретической части в рамках отчёта об итогах прослушанных лекций и проведённой работы кратко приведены полученные новые сведения о рассматриваемом контейнере.

1 Постановка задачи

Для прохождения учебной ознакомительной практики были поставлены следующие задачи:

1. Прослушать курс лекций, включающий в себя разбор следующих алгоритмов и структур данных:
    1) Вычислительная математика;
    2) Сортировка и бинарный поиск;
    3) Основы теории чисел;
    4) Структуры данных: set, map, vector, queue;
    5) Перебор;
    6) Графы;
    7) Динамическое программирование.
2. Решить несколько практических задач по каждой из вышеперечисленных тем для более полного понимания, как работают алгоритмы и как организовываются структуры данных.
3. Более подробно изучить одну из структур данных, а именно: неупорядоченный контейнер set.
4. Разработать несколько задач для иллюстрации работы структуры данных и провести сравнение с алгоритмами, использующими обычный упорядоченный set.
5. Сделать выводы о результатах проведенных исследований и сравнений.

2 Теоретическая часть

`Unordered_set` – это один из контейнеров библиотеки STL, а именно: ассоциативный неупорядоченный контейнер, так называемое неупорядоченное множество уникальных элементов. Он поддерживает итераторы только с прямым направлением, то есть, при переборе элементов `unordered_set` двигать итератор назад нельзя. Это связано с особенностью упорядочивания данных: такой контейнер представляет собой хеш-таблицу из ключей, которые определяются некоторой хеш-функцией. [2]

Организация данных таким способом предоставляет мгновенный доступ к данным по ключу, то есть операции добавления, удаления и поиска элемента в `unordered_set` требуют константное время. Это означает, что количество выполняемых операций не зависит от количества элементов в контейнере. [3]

Для использования контейнера необходимо подключить заголовочный файл `<unordered_set>`. Пример создания экземпляра класса `unordered_set`:

```
контейнер <тип_данных_контейнера> имя_переменной;
unordered_set <int>  s;
```

Ключ в хеш-таблице является константой и определяет позицию элемента, в связи с чем доступ к элементам осуществляется посредством итераторов. Итератор – это такая структура данных, которая используется для обращения к определенному элементу в контейнерах STL, аналог указателей, используемых при работе с массивом. Среди операций, реализуемых над итераторами, имеется инкремент и декремент `it++, it--`, а также функция `advance(iterator,value),` которая позволяет передвинуть итератор на указанное число позиций `value`. [4]

```
контейнер <тип_данных_контейнера> ::iterator <имя итератора>;
unordered_set <int> :: iterator it;
it = s.begin(); //итератор, указывающий на начало множества
advance(it,2);
```

Вставка элементов во множество `unordered_set` осуществляется с помощью метода `insert()` (при условии, что они уникальны для данного множества): [5]

```
s.insert(4);
s.insert(8);
for (auto x : s) cout << x << " "; //на выходе 4 8
```

Метод `erase()` позволяет удалить один элемент или элемент из диапазона. Удаление элементов на определённом диапазоне осуществляется с помощью итераторов, указывающих на границы диапазона: `s.erase(s.begin(), s.end())`. В случае одного элемента метод возвращает количество удалённых элементов:

```
unordered_set <int> s = { 1, 2, 3 };
cout << s.erase(1) << "\n"; // 1
cout << s.erase(1) << "\n"; // 0
 for (auto x : s) cout << x << ' '; // 3 2
```

Отдельно стоит выделить метод `emplace()`, который создает элемент и вставляет его во множество, если он уникален. Метод возвращает объект класса pair, и через свойство first можно получить доступ к итератору, указывающему на вставленный элемент, а через свойство second — получить логическое значение true, если элемент вставлен, или false — если он уже имеется во множестве. [5]

Среди других распространённых методов работы с контейнерами можно выделить следующие: [5]

- `size()` — возвращает количество элементов в контейнере;
- `empty()`— возвращает true, если множество пустое, иначе возвращает false;
- `clear()` — удаляет все элементы;
- `count(value)` — возвращает количество элементов со значением `value`;
- `find(value)` — возвращает итератор, установленный на элемент со значением `value`. Если элемент не найден, то метод возвращает итератор, указывающий на позицию после последнего элемента;
- `equal_range()` – возвращает экземпляр класса pair. Через свойство first будет доступен итератор, ссылающийся на первый элемент с указанным значением, а через свойство second — итератор, ссылающийся на позицию после последнего элемента с указанным значением. Если значение отсутствует, то оба итератора будут ссылаться на значение, возвращаемое методом `end()`.

3 Практическая часть

Задача 1. Дубликаты

Коллекционер-авантюрист собрал целых N карточек с изображением супергероев MARVEL. Карточки в коллекции могут повторяться больше двух раз или не повторяться вообще. Каждой карточке присвоен свой уникальный номер, выраженный натуральным числом. Зная номера всех карточек, необходимо найти среди них все дубликаты, если они есть, и исключить их из последовательности.

Входные данные. На вход подаётся число N ($1 \leq N \leq 10^5$) – количество чисел в наборе. Далее через пробел вводится последовательность чисел из N значений x_i ($0 \leq x_i \leq 10^6$).

Выходные данные. На первой строке необходимо вывести, сколько различных карточек имели дубликаты, а на второй строке – последовательность чисел без повторов элементов. Если дубликатов нет, вывести 0 на первой строке, а на второй – последовательность без изменений.

Пример 1.

input	output
8 5 7 1 1 8 1 6 5	2 5 7 1 8 6

Решение.

Для организации данных необходимо завести контейнер `unordered_set in` для хранения входных данных. На вход будут подаваться числа – номера карточек `x`, и наличие повторов будет определяться следующим условием: если метод `find(x)` не находит в последовательности введённое число `x` (то

есть указывает на конец контейнера), значит число уникальное и оно вводится в последовательность. В противном случае число является дубликатом, и его номер заносится в контейнер `duplicate`, причем искомое число повторов будет равняться числу элементов этого контейнера. Таким образом за один проход мы избавляемся от дубликатов карточек.

```
#include <iostream>
#include <unordered_set>
#include <vector>
#include <algorithm>
#include <set>
#include <fstream>
using namespace std;
int main()
{
	ifstream testinput("01");
	ofstream testoutput("01.a");
	unordered_set <int> in;
	unordered_set <int> duplicate;
	int n, x;
	testinput >> n;
	for (int i = 0; i < n; i++)
	{
		testinput >> x;
		if (in.find(x) == in.end()) in.insert(x);
		else
		{
			duplicate.insert(x);
		}
	}
	if (!duplicate.empty())
	{
		testoutput << duplicate.size();
	}
	else testoutput << 0;
	testoutput << "\n";
	for (auto z : in) testoutput << z << " ";
	testinput.close();
	testoutput.close();
	return 0; }
```

Тесты.

input	input	input
7	1	5
7 7 7 7 7 7 7	3	7 1 9 3 4
output	output	output
1	0	0
7	3	7 1 9 3 4

Тесты №1-3 иллюстрируют «крайние» случаи, когда все элементы последовательности одинаковые, или когда дубликатов нет совсем.
Тесты №4-20 проверяют алгоритм на существенно большем количестве вводимых данных, n до 10^5. Они будут приведены в отдельных файлах.

Замечание.

Использование неупорядоченного контейнера unordered_set будет более рациональным при большом количестве вводимых данных n, поскольку он не зависит от n и имеет ассимптотику O(1). Ниже приведено аналогичное решение с использованием простого контейнера set, в котором операции вставки в среднем занимают большее время, из-за чего алгоритм имеет оценку сложности O($\log n$).

В таблице №1 (Приложение А) приведены результаты сравнения времени работы алгоритмов. Замер времени осуществлялся с помощью функции `clock()`, в таблицу вносилось среднее значение из 5 замеров в секундах. Как видно из таблицы №1, unordered_set в реальной программе работает несколько быстрее обычного set.

Задача 2. Карточки

Коллекционер-авантюрист собрал всю коллекцию с героями MARVEL и, недолго думая, начал собирать новую. Теперь у него имеется много-много (целых N штук!) карточек с изображением супергероев DC. Каждой карточке присвоен свой уникальный номер, выраженный натуральным числом, однако сами карточки в коллекции могут повторяться. Коллекционер хочет быстро узнать, имеется ли в его коллекции карточка с определённым номером. Необходимо написать программу, которая будет отвечать на M таких запросов за O(1).

Входные данные. На вход подаётся число N ($1 \leq N \leq 10^4$) – количество чисел в наборе. Далее через пробел вводится последовательность чисел из N значений x_i ($0 \leq x_i \leq 10^6$). Числа в последовательности могут повторяться. На следующей строке вводится число M ($1 \leq M \leq 10^4$) – число запросов. Далее вводится M строк, на каждой – натуральное число x ($1 \leq x \leq N$) – номер карточки, наличие которой в коллекции нужно проверить.

Выходные данные. Необходимо вывести ответ на каждый вопрос в следующем формате:

А) Вывести «yes» (без кавычек), если запрошенный номер карточки встретился в коллекции хотя бы один раз;
Б) Вывести «no» (без кавычек) в случае, если запрошенный номер в коллекции отсутствует.

Пример 1.

input	output
8 5 7 1 1 8 1 6 5	yes

3 5 7 10	yes no

Решение.

Для хранения входных данных необходимо завести контейнер `unordered_set in`. Обработка поступающих запросов происходит в цикле for и осуществляется с помощью метода find(x). Если метод не находит в последовательности введённое число x (то есть указывает на конец контейнера), тогда выводится «no», в противном случае в консоль выводится «yes».

```
#include <iostream>
#include <unordered_set>
#include <vector>
#include <algorithm>
#include <set>
#include <fstream>
using namespace std;
int main()
{
	ifstream testinput("01");
	ofstream testoutput("01.a");
	unordered_set <int> in;
	unordered_set <int>::iterator init = in.begin();
	int n, m, x;
	testinput >> n;
	for (int i = 0; i < n; i++)
	{
		testinput >> x;
		in.insert(x);
	}
	testinput >> m;
	for (int i = 0; i < m; i++)
	{
		testinput >> x;
```

```
            if (in.find(x) == in.end()) testoutput << "no" << "\n";
            else testoutput << "yes" << "\n";
        }
        testinput.close();
        testoutput.close();
        return 0;
}
```

Тесты.

input	input	input
5	8	7
3 3 3 3 3	1 2 3 4 5 6 7 8	2 4 5 1 3 9 9
2	2	2
1	10	2
3	12	4
output	output	output
no	no	yes
yes	no	yes

Тесты №1-3 иллюстрируют «крайние» случаи, когда все элементы последовательности одинаковые, все элементы разные, когда ни на один из запросов нет ответа или когда имеется ответ на каждый.
Тесты №4-20 проверяют алгоритм на существенно большем количестве вводимых данных, n до 10^4. Они будут приведены в отдельных файлах.

Замечание.

При использовании абсолютно идентичного алгоритма с единственным лишь отличием в виде использования обычного контейнера set вместо unordered set ассимптотика становится $O(\log n)$, что при количестве запросов до 10^4 нерационально и может превышать время исполнения в 1 секунду. Именно этим фактом и обоснован выбор неупорядоченного хешированного контейнера unordered_set, в котором засчёт использования хэш-функции оценка сложности представляет собой O(1).

В таблице №2 (Приложение А) приведены результаты сравнения времени работы алгоритмов Замер времени осуществлялся с помощью функции `clock()`, в таблицу вносилось среднее значение из 5 замеров в секундах. Как видно из таблицы №2, unordered_set в реальной программе работает несколько быстрее обычного set.

Заключение

По итогам проведённой работы были изучены различные виды контейнеров стандартной библиотеки C++ под названием STL. Среди них, в частности, был подробно рассмотрен такой контейнер как unordered set, практическое применение которого было проиллюстрировано на примере самостоятельно разработанных и решённых задач.

Таким образом, на основе полученной информации можно сделать вывод о том, что рассматриваемый класс небезосновательно выбран в качестве основного объекта исследования: благодаря особенностям реализации с помощью хеш-таблиц контейнер unordered set позволяет быстро отвечать на запросы о поиске элементов внутри себя, а также добавлять элементы и удалять их.

В данном контексте быстрота означает мгновенный доступ к данным по ключу, то есть за константное время. Это позволяет не зависеть от количества элементов в контейнере, что делает данный контейнер полезным и нужным инструментом при работе с большим количеством запросов.

Библиографический список

1. STL: стандартная библиотека шаблонов C++ // TPROGER URL: https://tproger.ru/articles/stl-cpp/ (дата обращения: 18.07.2022).
2. Неупорядоченные множества в стандартной библиотеке шаблонов C++ // Computer Science Portal for geeks URL: http://espressocode.top/unordered_set-in-cpp-stl/ (дата обращения: 18.07.2022).
3. Контейнеры стандартной библиотеки C++ // C++ and Python URL: https://cpp-python-nsu.inp.nsk.su/textbook/sec3/ch3 (дата обращения: 18.07.2022).
4. Итераторы в C++: введение // codelessons URL: https://codelessons.ru/cplusplus/iteratory-v-c-vvedenie.html (дата обращения: 19.07.2022).
5. Учебник C++ (Qt Creator) / Классы unordered_set и unordered_multiset: множества // Прохоренок РФ URL: http://прохоренок.рф/pdf/cppmingw/ch18-cpp-klassy-unordered-set-unordered-multiset.html (дата обращения: 19.07.2022).

Приложение

ПРИЛОЖЕНИЕ А

input: N не менее 10^2 время работы set: 0.001 сек время работы unordered_set: 0.001 сек	input: N не менее 10^4 время работы set: 0.056 сек время работы unordered_set: 0.053 сек
input: N не менее 10^4 время работы set: 0.052 сек время работы unordered_set: 0.054 сек	input: N не менее 10^5 время работы set: 0.601 сек время работы unordered_set: 0.539 сек
input: N не менее 10^5 время работы set: 0.593 сек время работы unordered_set: 0.437 сек	input: N не менее 10^5 время работы set: 0.465 сек время работы unordered_set: 0.404 сек
input: N не менее 10^5 время работы set: 0.593 сек время работы unordered_set: 0.437 сек	input: N не менее 10^6 время работы set: 3.972 сек время работы unordered_set: 3.282 сек

Таблица №1. Результаты замеров времени работы программ для задачи №1.

input: N не менее 10, M не менее 10 время работы set: 0.002 сек время работы unordered_set: 0.001 сек	input: N не менее 10^2, M не менее 10^2 время работы set: 0.001 сек время работы unordered_set: 0.001 сек
input: N не менее 10^2, M не менее 10^2 время работы set: 0.002 сек время работы unordered_set: 0.001 сек	input: N не менее 10^2, M не менее 10^3 время работы set: 0.007 сек время работы unordered_set: 0.002 сек
input: N не менее 10^3, M не менее 10^3 время работы set: 0.013 сек время работы unordered_set: 0.011 сек	input: N не менее 10^3, M не менее 10^4 время работы set: 0.061 сек время работы unordered_set: 0.044 сек
input: N не менее 10^4, M не менее 10^4 время работы set: 0.058 сек время работы unordered_set: 0.043 сек	input: N не менее 10^4, M не менее 10^4 время работы set: 0.063 сек время работы unordered_set: 0.044 сек

Таблица №2. Результаты замеров времени работы программ для задачи №2.