C++

Разбор домашней работы

Итераторы

Что такое итератор?

Теория

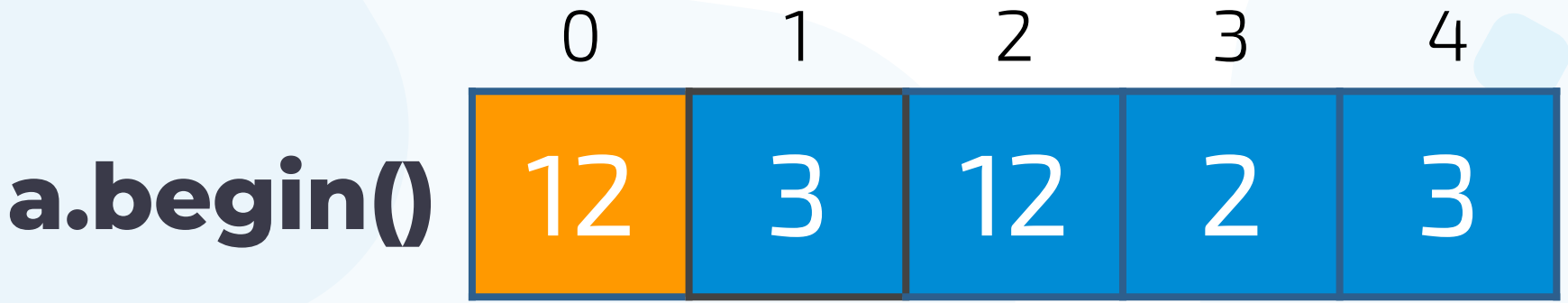
15 минут

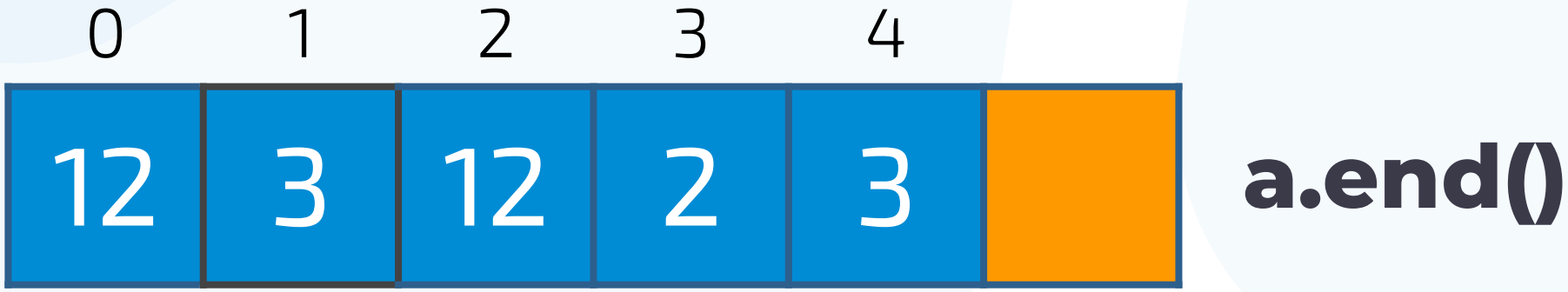
Итераторы

Итератор – объект, который способен перебирать элементы контейнера и осуществлять доступ к ним.

Функционал итераторов во многом совпадает с функционалом указателей.

Разберем итераторы на примере вектора.


0
1
2
3
4
a.begin()
12
3
12
2
3


0
1
2
3
4
12
3
12
2
3
a.end()

Итераторы

Итератор `a.begin()` дает нам указатель на первый элемент вектора.

Итератор `a.end()` дает указатель на элемент, следующий после последнего элемента вектора, т.е. сам `a.end()` не является уже частью вектора.

В какую переменную можно сохранить итератор?

Итераторы

Создание переменной, хранящей итератор на элемент вектора целых чисел:

```
vector<int>::iterator it = a.begin();
```

auto

auto – универсальный тип данных. Во время компиляции программы определяет необходимый тип данных.

```
auto it = a.begin();
```

Какие действия можно было делать с указателями?

Итераторы

Для того, чтобы получить итератор на элемент вектора, индекс которого нам известен, необходимо сдвинуть итератор, указывающий на первый элемент, на значение, равное индексу:

```
*(a.begin() + 3) = 4; //В элемент с индексом 3 будет записано значение 4
a[3] = 4; //Тот же результат
```

Пример

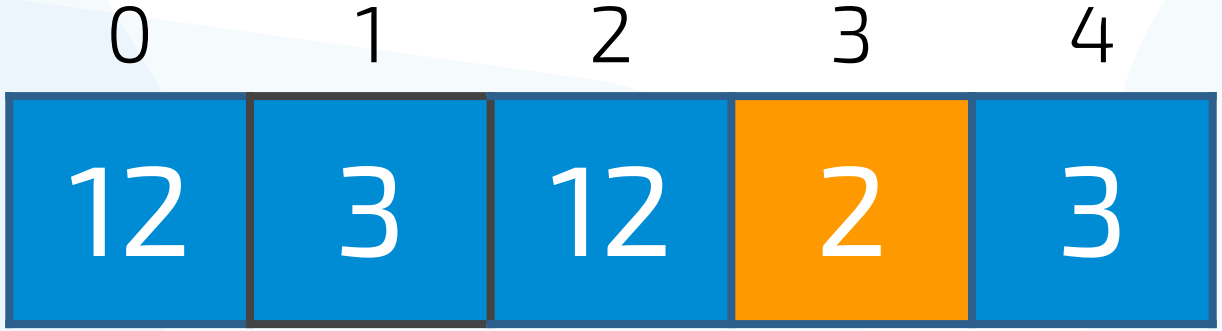
```
1.  int main() {
2.    vector<int> a(10);
3.  // Вводим значения в вектор
4.    for (auto it = a.begin(); it != a.end(); it++) {
5.      cin >> *it;
6.    }
7.    return 0;
8.  }
```

Вставка и удаление

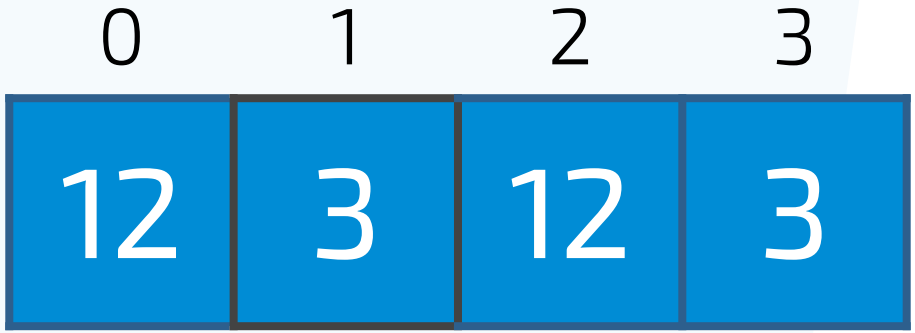
В отличие от строк, методы insert и erase требуют в качестве параметров не индексы позиции вставки или удаления соответственно, а итераторы на эти элементы вектора.

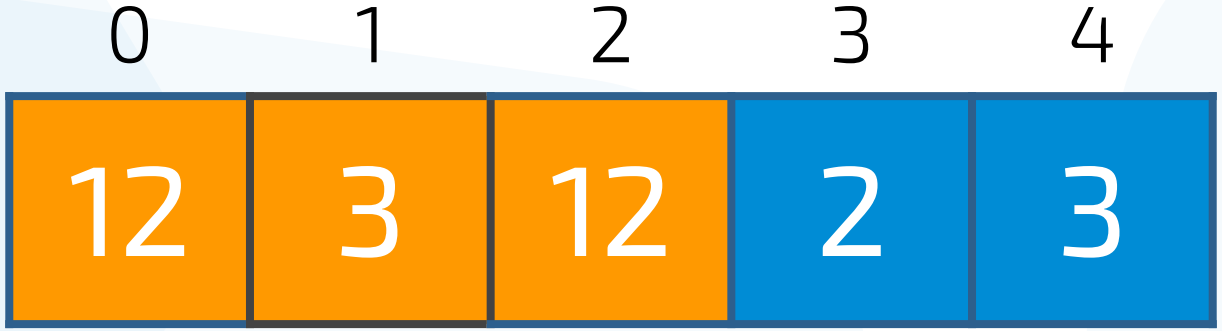
Пример

```
1.  int main() {
2.    vector<int> a;
3.  // Как-то заполняем вектор
4.    a.erase(a.begin() + 3);
5.  //Удаляется элемент с индексом 3
6.    a.erase(a.begin(), a.begin() + 3);
7.  //Удаляет первые 3 элемента вектора
8.    return 0;
9.  }
```

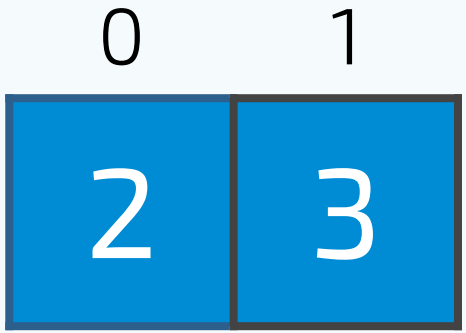



```
a.erase(a.begin() + 3)
```



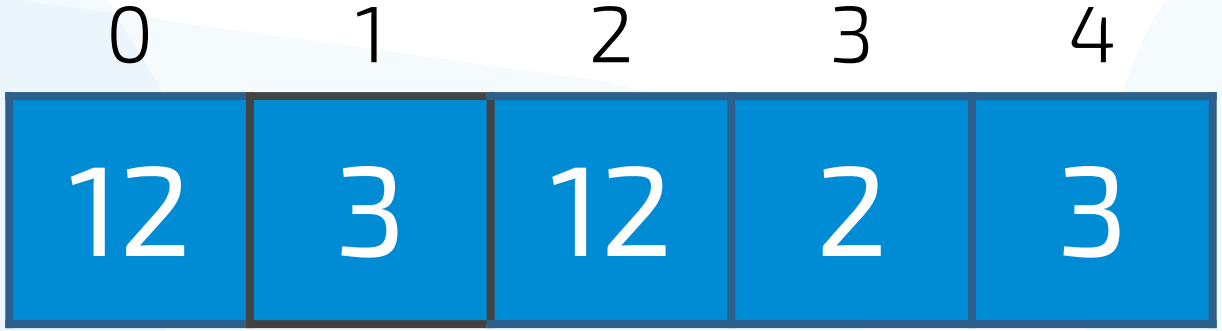



```
a.erase(a.begin(), a.begin() + 3)
```

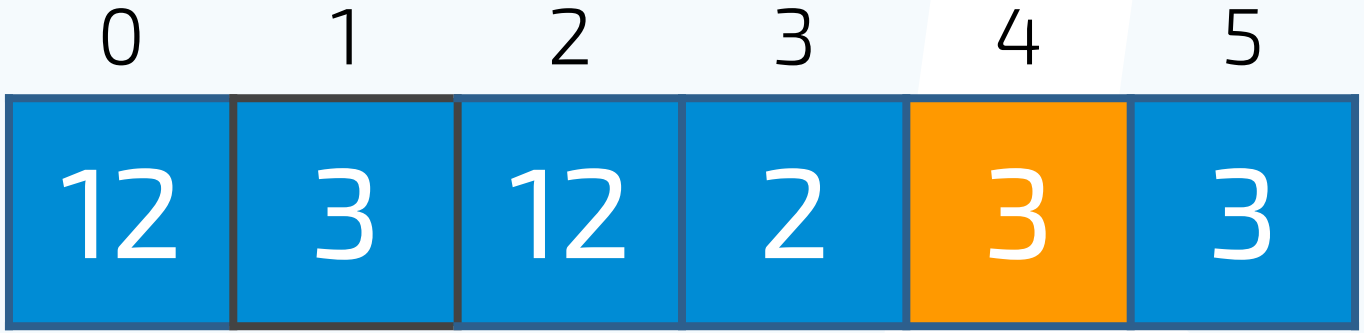


Пример

```
1.  int main() {
2.    vector<int> a;
3.  // Как-то заполняем вектор
4.    a.insert(a.begin() + 3, 4);
5.  /*Вставляет значение 4 на место элемента
6.  с индексом 3, сдвигая остальные элементы вправо*/
7.    return 0;
8.  }
```




```
a.insert(a.begin() + 4, 3)
```



Практика

15 минут

Теория

10 минут

<algorithm>

Разберем некоторые полезные функции, которые могут работать с массивами, но как параметры принимают указатели (а некоторые и как результат также возвращают указатели)

Функции <algorithm>

sort(итератор на начало, итератор на конец)
Данная функция сортирует массив в заданном промежутке по неубыванию. Пример:

```
sort(a.begin(), a.end());
// отсортирует все элементы вектора
```

is_sorted(начало, конец)
Функция, проверяющая, отсортирован ли заданный диапазон по неубыванию. Как результат возвращает `true` или `false`

Функции <algorithm>

reverse(итератор на начало, итератор на конец)

Данная функция переворачивает последовательность в заданном диапазоне:

```
reverse(a.begin(), a.end());
// перевернет вектор
```

count(начало, конец, значение)

Функция, определяющая количество элементов в заданном диапазоне, равных указанному значению

Функции <algorithm>

replace(начало, конец, старое значение, новое значение)
Данная функция заменяет все элементы, равные некоему значению, на новое значение. Пример:

```
replace(a.begin(), a.end(), 1, 2);
// заменит все единицы на двойки
```

equal(начало первого диапазона, конец первого диапазона, начало второго диапазона)
Функция, определяющая, равны ли два заданных диапазона. Для второго диапазона не задается конец, потому что предполагается, что они должны быть одинаковых размеров.

Функции <algorithm>

min_element(начало, конец)

Данная функция возвращает указатель на первый минимальный элемент в заданном диапазоне. Чтобы получить само значение, указатель надо разыменовать:

```
int min = *min_element(a.begin(), a.end()); //Значение
int min_ind = min_element(a.begin(), a.end()) - a.begin();
// Индекс
```

max_element(начало, конец)

Аналогично, только определяет первый максимальный элемент.

Практика

10 минут

Теория

10 минут

Какие еще арифметические действия можно делать с указателями и итераторами?

Разность указателей и итераторов

Итераторы, как и указатели, можно вычитать друг из друга.

Как результат будет получаться расстояние между ячейками памяти, поэтому если вычесть из итератора на элемент вектора, например, итератор на начало вектора, то можно получить индекс такого элемента.

Пример

```
1.  int main() {
2.    vector<int> a;
3.  // как-то заполняем вектор
4.    auto it = min_element(a.begin(), a.end());
5.  // находим итератор на минимальный элемент
6.    cout << it - a.begin();
7.  // выводим индекс минимального элемента
8.    return 0;
9.  }
```

Практика

10 минут

Итоги урока

1) Что такое итератор
2) Создание переменной для хранения итератора
3) Вставка и удаление в векторе
4) Алгоритмы
5) *Разность итераторов и указателей