|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название класса** | **Пара (pair)** | **Массив (array)** | **Вектор (vector)** | **Список (list)** |
| Какую библиотеку подключить | <utility> | <array> | <vector> | <list> |
| Синтаксис инициализации | std::pair <std::string, std::string> myName(“Ivan”, “Ivanov”); | std::array <тип массива, размер массива>  std::array <int, 5> M;  std::array <int, 5> M = {1,2,3,4,5}; | std::vector <тип данных вектора> идентификатор  std::vector<int> V1= {2,5};  std::vector<int> V2 {5,1} | std::list<тип данных> идентификатор  std::list<int> listI = {1,2,3,4}; |
| Какими способами можно получить доступ к произвольному элементу класса? | В паре может быть всего два элемента. Доступ получается с помощью .first для первого элемента и .second для второго. Например: myName.first = “Ivan”, myName.second = “Ivanov”. | Через цикл с использование индекса  for(int i = 0; i < 5; i++)  {  std::cout << M[i] << “\_\_“;  }  Через функцию at(); | Через оператор [i]  V1[1] = 5;  и через функцию at(); | .front() и .back() возвращает первый и последний элемент списка;  listM.front(); // 1  listM.back(); // 4  С помощью цикла foreach:  for(int n : listM)  {  std::cout << n << endl;  } |
| Можно ли добавлять новые элементы? Если да, то каким образом? С какой стороны разрешено их добавлять? | Нельзя. | Нельзя, это фиксированный массив, можно только изменять элементы по индексу. | Это можно сделать с помощью методов push\_back(), insert()  push\_back() — добавляет элемент в конец;  insert() – добавляет элемент в любую позицию | Можно.  push\_back(val): добавляет значение val в конец списка  push\_front(val): добавляет значение val в начало списка  insert(pos, val): вставляет элемент val на позицию, на которую указывает итератор pos, аналогично функции emplace. Возвращает итератор на добавленный элемент  и другие функции |
| Можно ли удалять элементы? Если да, то каким образом? С какой стороны разрешено их удалять? | Нельзя. | Нельзя, это фиксированный массив, можно только изменять элементы по индексу. | Для этого используется метод erase(). Этот метод удаляет элемент по индексу, либо диапазон элементов по индексам. | Можно.  clear(p): удаляет все элементы  pop\_back(): удаляет последний элемент  pop\_front(): удаляет первый элемент  erase(p): удаляет элемент, на который указывает итератор p. Возвращает итератор на элемент, следующий после удаленного, или на конец контейнера, если удален последний элемент |
| Каким способом можно узнать количество элементов в классе? | Класс «пара» всегда подразумевает наличие двух элементов. | size() – возвращает размер контейнера | Для получения размера массива vector можно использовать функцию size(). | Для получения размера списка можно использовать функцию size(). |
| Предлагает ли класс автоматическую сортировку данных? Если да, то какую? | Не предлагает. | Не предлагает | Не предлагает | Не предлагает |
| При каком условии объект этого класса может содержать два или более одинаковых значений? | Всегда может. | При любых условиях | При любых условиях | При любых условиях |
| Какие специфичные функции можно применять к классу? Что они делают? (только из того, что мы изучали на лекции) | Функция std::make\_pair(value, value) позволяет быстро создавать пары из двух значений. | fill(n): присваивает всем элементам контейнера значение n. | push\_back() — добавляет элемент в стек;  back() — возвращает значение верхнего элемента стека;  pop\_back() — вытягивает элемент из стека. | push\_back(val): добавляет значение val в конец списка  push\_front(val): добавляет значение val в начало списка |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название класса** | **Односвязный список (forward list)** | **Дек (deque)** | **Словарь (map/multimap)** | **Набор (set/multiset)** |
| Какую библиотеку подключить | <forward\_list> | <deque> | <map> | <set> |
| Синтаксис инициализации | std::forward\_list <тип данных> идентификатор  std::forward\_list<int> list1 {1,2,3}  std::forward\_list<int> list2 = {1,2,3,4,5}  std::forward\_list<int> list3(1,2) | std::deque <тип данных> идентификатор  std::deque deque1(5, 2);  std::deque deque2{ 1, 2, 4, 5 }; std::deque deque3 = { 1, 2, 3, 5 }; | std::map<тип ключа, тип хранения> идентификатор;  map<string, int> myMap = { {“Darya”, 33}, {“Artem”, 20} }  multimap <char, int> myMultimap  myMultimap.insert(pair<char,int> (‘a’,1)); | std:: set<тип данных> идентификатор  std::set<int> s;  std::multiset<int> ms; |
| Какими способами можно получить доступ к произвольному элементу класса? | Напрямую в списке forward\_list можно получить только первый элемент. Для этого применяется функция front().  int first = list1.front();  Для перебора элементов также можно использовать цикл:  for (int n : numbers)  {  std::cout << n << "\_\_";  } | [index]: получение элемента по индексу  at(index): возращает элемент по индексу  front(): возвращает первый элемент  back(): возвращает последний элемент | begin() - итератор на первый элемент;  end() - итератор на элемент идущий после последнего;  find(key) - итератор на первый элемент с указанным ключом;  lower\_bound(key) - итератор на первый элемент, чей ключ больше или равен указанному ключу;  upper\_bound(key) - итератор на первый элемент, чей ключ больше указанного ключа;  equal\_range(key) - диапазон элементов, чей ключ равен указанному ключу;  [index] - операция индексации по ключу | S.find(a) - найти элемент a в контейнере,  S.lower\_bound(a) - первый элемент, не меньший чем a,  S.upper\_bound(a) - первый элемент, больший чем a,  S.equal\_range(a) - пара элементов, первый - нижняя граница элементов с, такими же значениями, что и a, второй - верхняя граница элементов с такими же значениями, что и a.  S.begin() - указатель на начало контейнера,  S.end() - указатель на конец контейнера,  S.rbegin() - реверсивный указатель на конец контейнера,  S.rend() - реверсивный указатель на начало контейнера |
| Можно ли добавлять новые элементы? Если да, то каким образом? С какой стороны разрешено их добавлять? | push\_front(val): добавляет объект val в начало списка  emplace\_front(val): добавляет объект val в начало списка  emplace\_after(p, val): вставляет объект val после элемента, на который указывает итератор p. Возвращает итератор на вставленный элемент. Если p представляет итератор на позицию после конца списка, то результат неопределен.  insert\_after(p, val): вставляет объект val после элемента, на который указывает итератор p. Возвращает итератор на вставленный элемент.  insert\_after(p, n, val): вставляет n объектов val после элемента, на который указывает итератор p. Возвращает итератор на последний вставленный элемент.  insert\_after(p, begin, end): вставляет после элемента, на который указывает итератор p, набор объектов из другого контейнера, начало и конец которого определяется итераторами begin и end. Возвращает итератор на последний вставленный элемент.  insert\_after(p, il): вставляет после элемента, на который указывает итератор p, список инициализации il. Возвращает итератор на последний вставленный элемент. | push\_front(val): добавляет объект val в начало списка  emplace\_front(val): добавляет объект val в начало списка  emplace\_after(p, val): вставляет объект val после элемента, на который указывает итератор p. Возвращает итератор на вставленный элемент. Если p представляет итератор на позицию после конца списка, то результат неопределен.  insert\_after(p, val): вставляет объект val после элемента, на который указывает итератор p. Возвращает итератор на вставленный элемент.  insert\_after(p, n, val): вставляет n объектов val после элемента, на который указывает итератор p. Возвращает итератор на последний вставленный элемент.  insert\_after(p, begin, end): вставляет после элемента, на который указывает итератор p, набор объектов из другого контейнера, начало и конец которого определяется итераторами begin и end. Возвращает итератор на последний вставленный элемент.  insert\_after(p, il): вставляет после элемента, на который указывает итератор p, список инициализации il. Возвращает итератор на последний вставленный элемент. | insert(el) - вставка элемента, возвращается его позиция;  insert(beg,end) - вставка элементов из указанного диапазона; | insert(a) - вставка элемента a в конец множества |
| Можно ли удалять элементы? Если да, то каким образом? С какой стороны разрешено их удалять? | pop\_front(): удаляет первый элемент  erase\_after(p): удаляет элемент после элемента, на который указывает итератор p. Возвращает итератор на элемент после удаленного  erase\_after(begin, end): удаляет диапазон элементов, на начало и конец которого указывают соответственно итераторы begin и end. Возвращает итератор на элемент после последнего удаленного | clear(p): удаляет все элементы  pop\_back(): удаляет последний элемент  pop\_front(): удаляет первый элемент  erase(p): удаляет элемент, на который указывает итератор p. Возвращает итератор на элемент, следующий после удаленного, или на конец контейнера, если удален последний элемент  erase(begin, end): удаляет элементы из диапазона, на начало и конец которого указывают итераторы begin и end. Возвращает итератор на элемент, следующий после последнего удаленного, или на конец контейнера, если удален последний элемент | erase(key) - удалить указанный элемент;  erase(it), erase(start,end) - удаляет элемент с заданным итератором или между заданными  clear() - удалить все элементы. | используется функция erase(), которая принимает ключ элемента, который нужно удалить. |
| Каким способом можно узнать количество элементов в классе? | max\_size() – возвращает размер контейнера | size() – возвращает размер контейнера | size() - возвращает число элементов;  max\_size() - максимально возможный размер отображения;  count(key) - число элементов соответствующих указанному ключу | size() - возвращает число элементов;  max\_size() - максимально возможный размер отображения;  count() - число элементов; |
| Предлагает ли класс автоматическую сортировку данных? Если да, то какую? | Не предлагает | Не предлагает | Не предлагает | Не предлагает |
| При каком условии объект этого класса может содержать два или более одинаковых значений? | Повторяющиеся элементы могут быть вставлены с помощью функции push\_back() | Повторяющиеся элементы могут быть вставлены с помощью функции push\_back() | Повторяющиеся элементы возможны, если ключ имеет повторяющееся значение | Повторяющиеся элементы возможны, если ключ имеет повторяющееся значение |
| Какие специфичные функции можно применять к классу? Что они делают? (только из того, что мы изучали на лекции) | assign(il): заменяет содержимое контейнера элементами из списка инициализации il  assign(n, value): заменяет содержимое контейнера n элементами, которые имеют значение value  assign(begin, end): заменяет содержимое контейнера элементами из диапазона, на начало и конец которого указывают итераторы begin и end | assign(il): заменяет содержимое контейнера элементами из списка инициализации il  assign(n, value): заменяет содержимое контейнера n элементами, которые имеют значение value  assign(begin, end): заменяет содержимое контейнера элементами из диапазона, на начало и конец которого указывают итераторы begin и end | (\*iter).first — для обращения к ключу и (\*iter).second -для обращения к данным,  где iter - итератор элемента | S.lower\_bound(a) - первый элемент, не меньший чем a,  S.upper\_bound(a) - первый элемент, больший чем a,  S.equal\_range(a) - пара элементов, первый - нижняя граница элементов с, такими же значениями, что и a, второй - верхняя граница элементов с такими же значениями, что и a. |