ОСНОВЫ C++
STL

Указатели и ссылки

- int a;
- int *b = &a;
  b = NULL;
  b = &a1;
  *b = 1;
- int &c = a;
  c = 1;

Передача параметров

- По значению:
  void f(int x);
  f(1); f(a);
- По ссылке:
  void f(int& x);
  f(a);
- По константной ссылке:
  void f(const int& x)
  f(1); f(a);

Функции-члены

- struct intArray {
  ```
  int m_a[N];
  void set(int idx, int v) { /* void f(S* this, int y) */
      m_a[idx] = v;
  }
  int get(int idx) {
      return m_a[idx];
  }
  ```
  };
- intArray a, *b = &a;
  a.set(0, 42);
  cout << b->get(0);

Перегрузка операторов

- struct intArray {
  int _a[N];
  int& operator[](int idx) { return a[idx]; }
  intArray operator+(const intArray& other) const {
  intArray ret;
  for (int i = 0; i < N; ++i) {
  ret._a[i] = _a[i] + other._a[i];
  }
  return ret;
  }
  };
- intArray a;
  a[0] = 42;
  cout << a[0]

Функторы

- struct F {
  ```
  double k;
  F(double _k) { k = _k; }
  double operator()(double x) { return sin(k * x); }
  ```
  }
- F f(2);
  double sin2pi = f(M_PI);

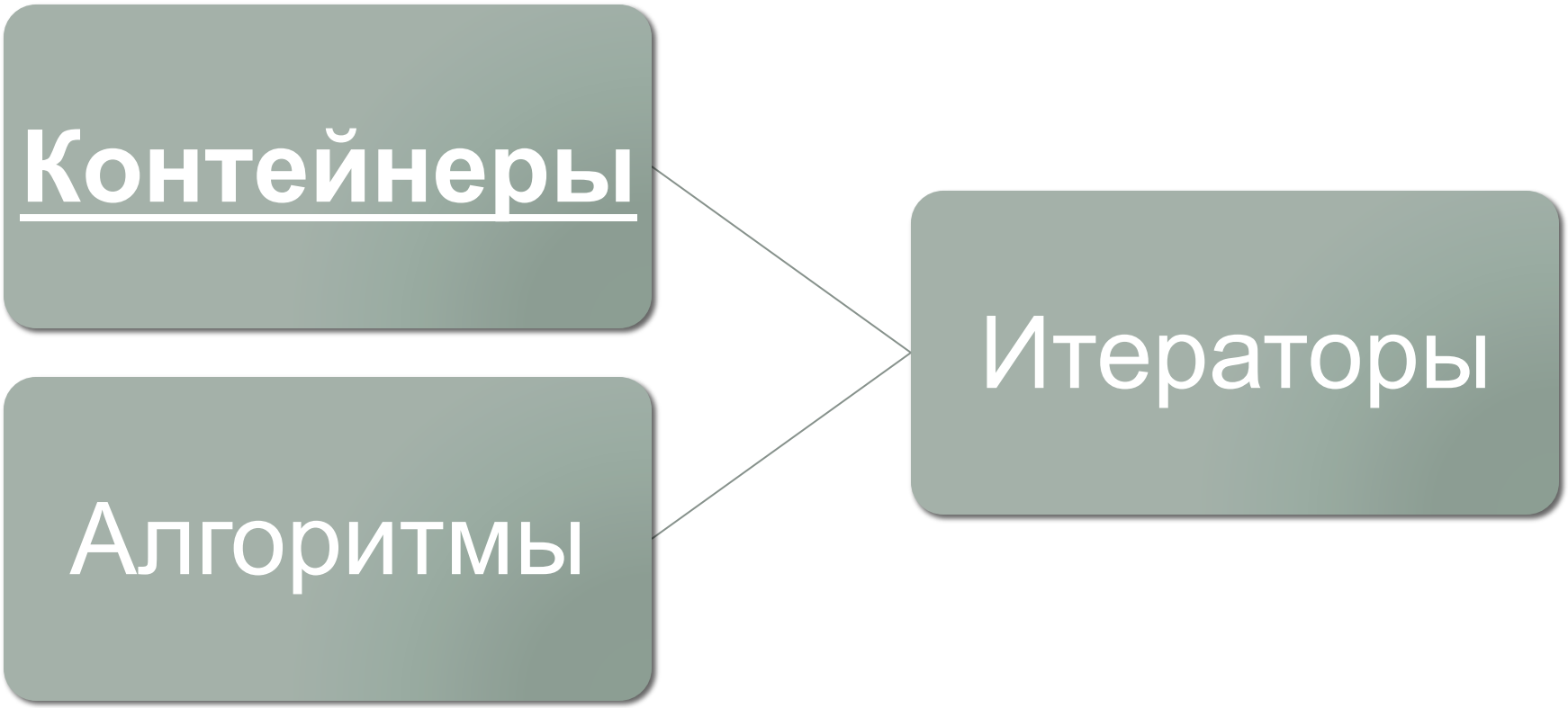
Шаблоны

- ```
  template<typename T, int N>
  struct Array {
          T _a[N];
          T& operator[](int idx) { return a[idx]; }
          Array operator+(const Array<T, N>& other) const {
                  Array<T, N> ret;
                  for (int i = 0; i < N; ++i) {
                          ret._a[i] = _a[i] + other._a[i];
                  }
                  return ret;
          }
  };
  ```
- ```
  Array<int, 10> x, y;
  x = x + y;
  ```

<utility>

- template<typename T, typename U>
  struct std::pair {
  T first;
  U second;
  };
  template<typename T, typename U>
  pair<T, U> make_pair(const T& first, const U& second);
- template<typename T>
  struct std::less {
  bool operator()(const T& a, const T& b) {
  return a < b;
  }
  };

STL



std::vector<T>

- T& operator[](int) /* O(1), NI */
- void push_back(const T&) /* O(1), I */
- iterator erase(iterator) /* O(N), I */
- iterator insert(iterato, const T&) /* O(N), I */
- iterator begin(); intetaror end(); /* O(1), NI */

- Занимаемая память: N * sizeof(T) + A,
  A >= 3 * sizeof(void*)
- При T = bool: N * sizeof(T) / 8 + A
  - Но operator[] возвращает не bool&
- При T != bool, vector<T>::iterator == T*

std::deque<T>

- T& operator[](int) /* O(1), NI */
- void push_back(const T&) /* O(1), NI */
- void push_front(const T&) /* O(1), NI */
- void pop_back(); void pop_front() /* O(1), NI */
- iterator erase(iterator) /* O(N), I */
- iterator insert(iterato, const T&) /* O(N), I */
- iterator begin(); iterator end(); /* O(1), NI */

- Занимаемая память: N * (A + sizeof(T)) + B,
  A * sizeof(T) ~ 4k, B >= 4 * sizeof(void*)

std::queue<T>, std::stack<T>

- void queue::push(const T&)
- T& queue::front()
- void queue::pop()

- void stack::push(const T&)
- T& stack::top()
- void stack::pop()

std::priority_queue<T, Container, Comp>

- Container = std::vector<T>, Comp = std::less<T>

- void pop(); /* O(ln N) */
- T& top(); /* O(1) */
- void push(const T&); /* O(ln N) */

- Использование памяти: как у vector<T>

std::list<T>

- void push_back(const T&) /* O(1), NI */
- void push_front(const T&) /* O(1), NI */
- void pop_back(); void pop_front() /* O(1), NI */
- void splice(iterator pos, list& x, iterator f, iterator s); /*O(1)/O(N), NI */
- int size() const /* O(N)/O(1), NI */
- iterator begin(); iterator end(); /* O(1), NI */

- Занимаемая память: (sizeof(T) + A) * N + B, A >= 2*sizeof(void*), B >= 3*sizeof(void*)

std::set<T, Comp>

- Comp = std::less<T>

- void insert(const T&) /* O(ln N), NI */
- iterator find(const T&) /* O(ln N), NI */
- iterator lower_bound(const T&)
- iterator upper_bound(const T&)
- pair<iterator, iterator> equal_range(const T&)

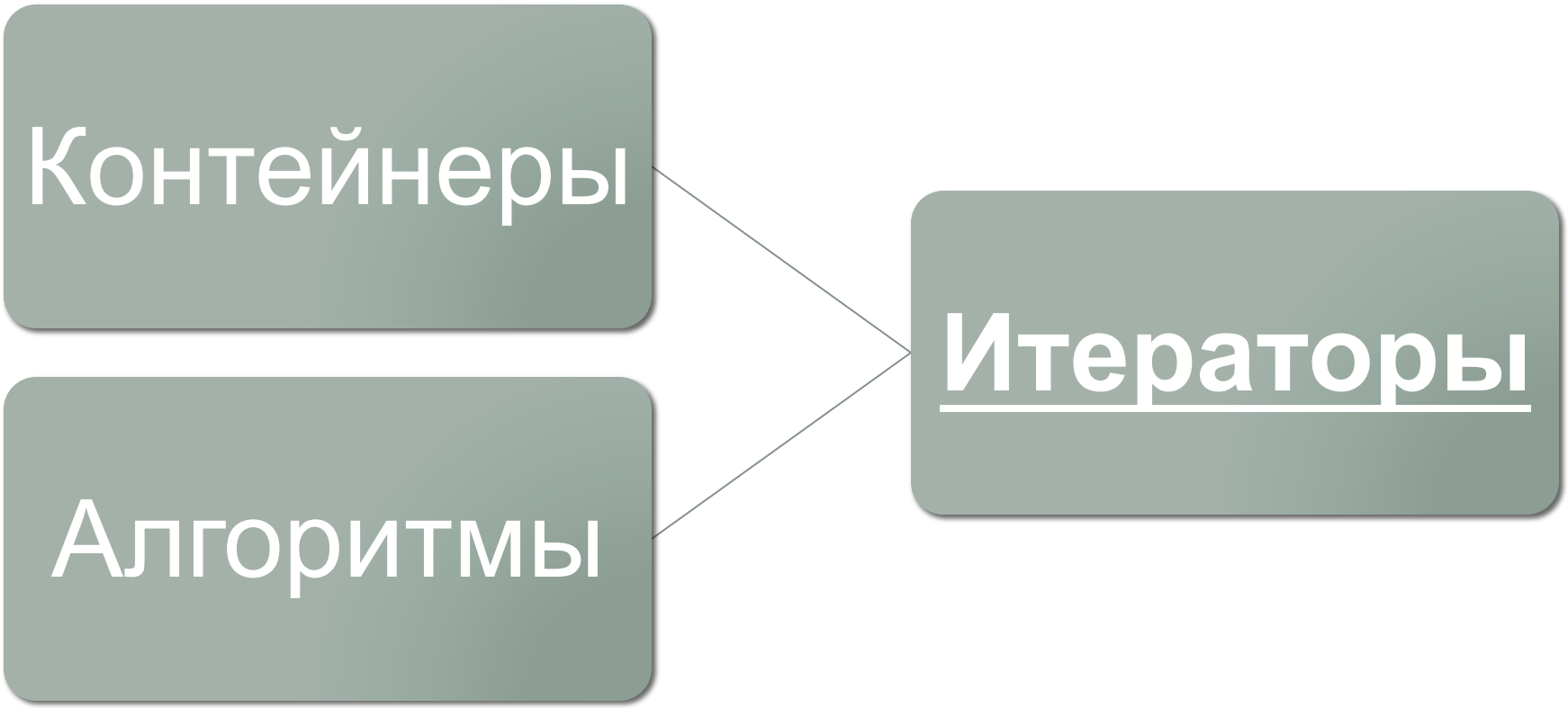
- Использование памяти: (A + sizeof(T)) * N + B,
  A >= 2 * sizeof(void*), B >= sizeof(void*) + sizeof(int)

std::map<Key, T, Comp>

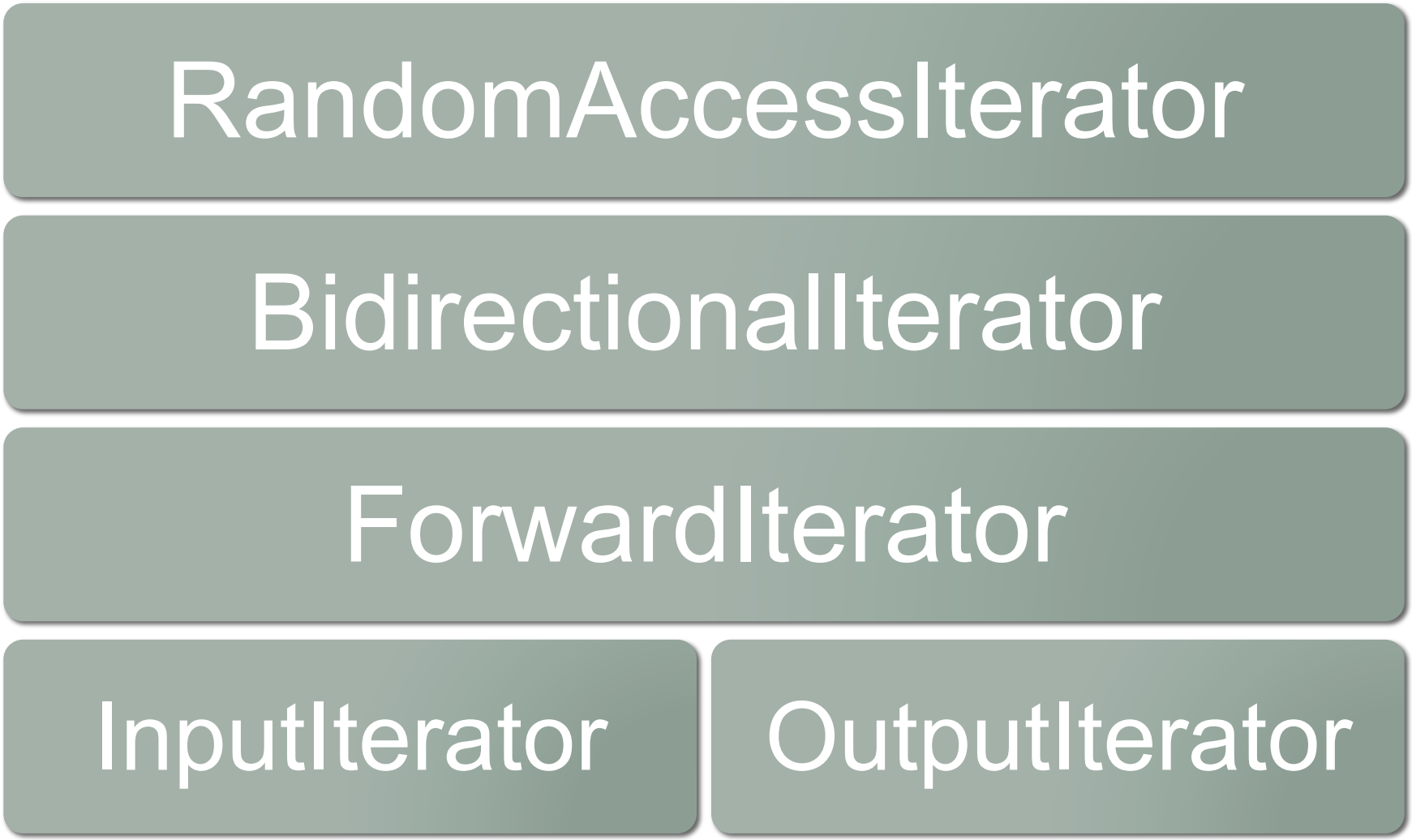
- Comp = std::less<Key>

- Value& operator[](const Key& k) /* O(ln N), NI */
- typedef pair<Key, T> value_type;

STL



iterator

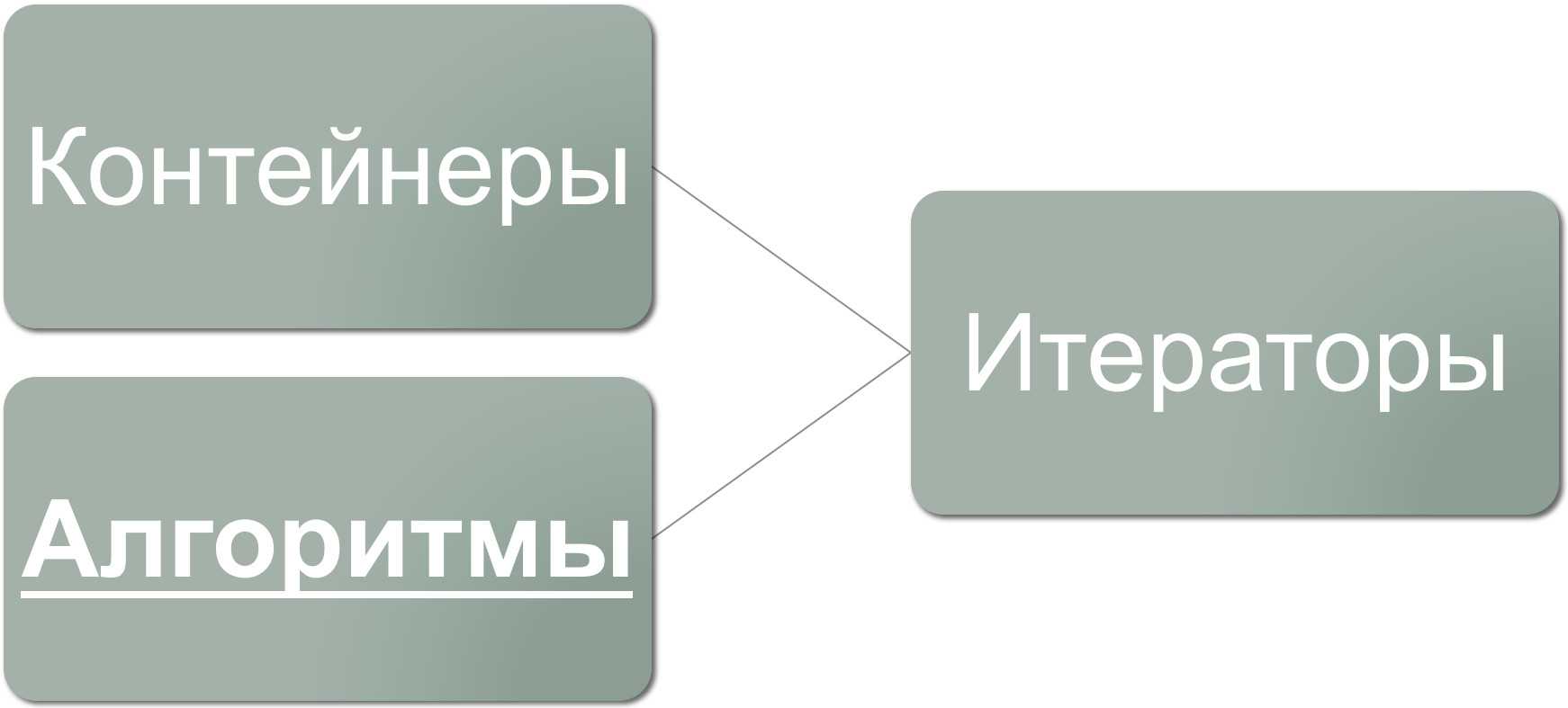

RandomAccessIterator
BidirectionalIterator
ForwardIterator
InputIterator
OutputIterator

Использование итераторов

- template<typename T>
  void print_container(const T& a) {
  for (T::iterator i = a.begin(); i != a.end(); ++i) {
  cout << (*i);
  }
  }

- vector<int> x;
  print_container(x);

STL



std::sort

- template<typename T, typename Comp>
  std::sort(RandomAccessIterator begin,
  RandomAccessIterator end[,
  Comp cmp]);
- bool intsLess(int a, int b) { return a > b; }
  sort(a, a + N, intsLess);
- struct intsLess() {
  bool operator()(int a, int b) { return a > b; }
  }
  sort(a, a + N, intsLess());

RandomAccessIterator

- random_shuffle(RndIt begin, RndIt end)
  /* O(N) */
- stable_sort(RndIt begin, RndIt end, [Comp cmp])
  /* O(N*ln N) */
- nth_element(RndIt begin, RndIt nth, RndIt end, [cmp])
  /* O(N) */
- RndIt lower_bound(RndIt begin, RndIt end, const T& x)
  RndIt upper_bound(RndIt begin, RndIt end, const T& x)
  RndIt binary_search(RndIt begin, RndIt end, const T& x)
  pair<RndIt, RndIt> equal_range(…)
  /* O(ln N), требуют отсортированный контейнер */

{Input,Output}Iterator

- bool contains(IIt begin1, IIt end1, IIt begin2, IIt end2)
- OutIt merge(IIt begin1, IIt end1, IIt begin2, IIt end2, OIt o)
- OutIt set_difference(…)
- OutIt set_intersection(…)
- OutIt set_symmetric_difference(…)
- OutIt set_union(…)

- /* O(N), требуют отсортированный контейнер */

BidirIterator

- bool next_permutation(BidirIt begin, BidirIt end, Comp);
- bool prev_permutation(BidirIt begin, BidirIt end, Comp);

Задача из ЕГЭ

- struct Cmp { bool operator()(const pair<…>& p1, const pair<…>& p2) {
  return p1.second > p2.second;
  };
  map<string, int> votes;
  int n, k; cin >> n >> k;
  for (int i = 0; i < n; ++i) {
  string cur; cin >> cur;
  votes[cur]++;
  }
  vector<pair<string, int> > results;
  results.reserve(votes.size());
  for (map<string, int>::iterator i = votes.begin(); i != votes.end(); ++i) {
  results.push_back(make_pair(i->first, i->second));
  }
  kth_element(results.begin(), results.end(), results.begin() + k, Cmp());
  stable_sort(results.begin(), results.begin() + k, Cmp());