Примеры.

Поиск анаграмм

- Поиск в словаре всех анаграмм данного слова

Сеанс работы

```
Программа поиска анаграмм:
поиск в словаре всех слов, которые могут
быть образованы из букв данного слова.

Введите имя файла со словарем: diction

Чтение словаря...
Словарь содержит 20159 слов.

Введите слово (или строку букв): rseuce
cereus
recuse
rescue
secure

Введите другое слово
(или символ конца файла для останова): nadeirga
drainage
gardenia

Введите другое слово
(или символ конца файла для останова): grinch
Ничего не найдено.

Введите другое слово
 (или символ конца файла для останова):
```

Программа поиска анаграмм

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;

int main()
{
    cout << "Программа поиска анаграмм:\n"
         << "поиск в словаре всех слов, которые могут\n"
         << "быть образованы из букв данного слова.\n"
         << endl;

    <Получение имени словаря и подготовка к чтению 211а>
    <Копирование словаря в вектор 211б>
    <Запрос слов и поиск анаграмм 212а>

  return 0;
}
```

Получение имени словаря и подготовка к чтению

```
cout << "Find of anagram"<< endl;
cout << "Input file name: " << flush;
string dictionary_name;
cin >> dictionary_name;
ifstream ifs(dictionary_name.c_str());

if (!ifs.is_open())
{
        cout << "Not found such file:" << dictionary_name << endl;
        exit(1);
}
```

Копирование словаря в вектор

```
cout << "Read the dictionary..." << flush;

typedef istream_iterator<string> string_input;
vector<string> dictionary;

copy(string_input(ifs), string_input(),back_inserter(dictionary));

cout << "dictionary size " << dictionary.size()<< endl;;
```

```
    cout << "Input word: " << flush;
    for (string_input j(cin); j != string_input(); ++j)          {
            string word = * j ;
            sort (word.begin () , word.end());
            bool found_one = false;
            do {
                if (binary_search(dictionary.begin(),dictionary.end(),word)) {
                        cout << " " << word << endl;
                        found_one = true;
                }
            } while (next_permutation(word.begin(), word.end()));
            if (!found_one)
                        cout << "Not found";
            cout << "Input word or Ctrl+Z"<< endl;
    }
    return 0;
}
```

Поиск анаграмм

- Поиск всех групп анаграмм в словаре

Внутренний словарь в виде вектора, хранящий пары строк

- второй член пары хранит слово w из файла словаря;
- первый член хранит строку, которая представляет собой результат сортировки букв w в алфавитном порядке.

```
argon cater cereus groan maker organ secure trace
```

first	second
agnor	argon
acert	cater
ceersu	cereus
agnor	groan
aekmr	maker
agnor	organ
ceersu	secure
acert	trace

first	second
acert	cater
acert	trace
aekmr	maker
agnor	argon
agnor	groan
agnor	organ
ceersu	cereus
ceersu	secure

```
// Anagram2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include <iterator>
#include <functional>
using namespace std;
```

Определение структуры данных

```
struct PS : pair<string, string>
{
        PS () : pair<string, string>(string(), string()) { }
        PS(const string& s) : pair<string, string>(s, s)
        {
                sort(first.begin(), first.end());
        }
        operator string() const { return second; }
};
```

Создание функциональных объектов для сравнений

```
struct FirstLess : binary_function<PS, PS, bool>
{
        bool operator()(const PS& p, const PS& q) const
        {
                return p.first < q.first;
        }
} firstLess;


struct FirstEqual : binary_function<PS, PS, bool>
{
        bool operator()(const PS& p, const PS& q) const
        {
                return p.first == q.first;
        }
} firstEqual;
```

```
int main() {
    cout << "Программа поиска групп анаграмм:\n"
         << "находит все группы анаграмм в словаре.\n\n"
         << flush;

<Получение имени словаря и подготовка к чтению 211а>
<Копирование словаря в вектор объектов PS 220б>
<Сбор всех групп анаграмм 221а>
<Подготовка к выводу групп анаграмм 221б>
<Вывод всех групп анаграмм 221в>

  return 0;
}
```

```
cout << "Find of anagram"<< endl;
        cout << "Input file name: " << flush;
        string dictionary_name;
        cin >> dictionary_name;
        ifstream ifs(dictionary_name.c_str());
        if (!ifs.is_open())
        {
                cout << "Not found such file:" << dictionary_name << endl;
                exit(1);
        }
```

Чтение словаря в вектор объектов

```
cout << "Read the dictionary..." << flush;
typedef istream_iterator<string> string_input;

vector<PS> word_pairs;
copy(string_input(ifs), string_input(), back_inserter(word_pairs));
cout << "Find from " << word_pairs.size() << " anagram group..." << flush;
```

Использование объекта сравнения для сортировки

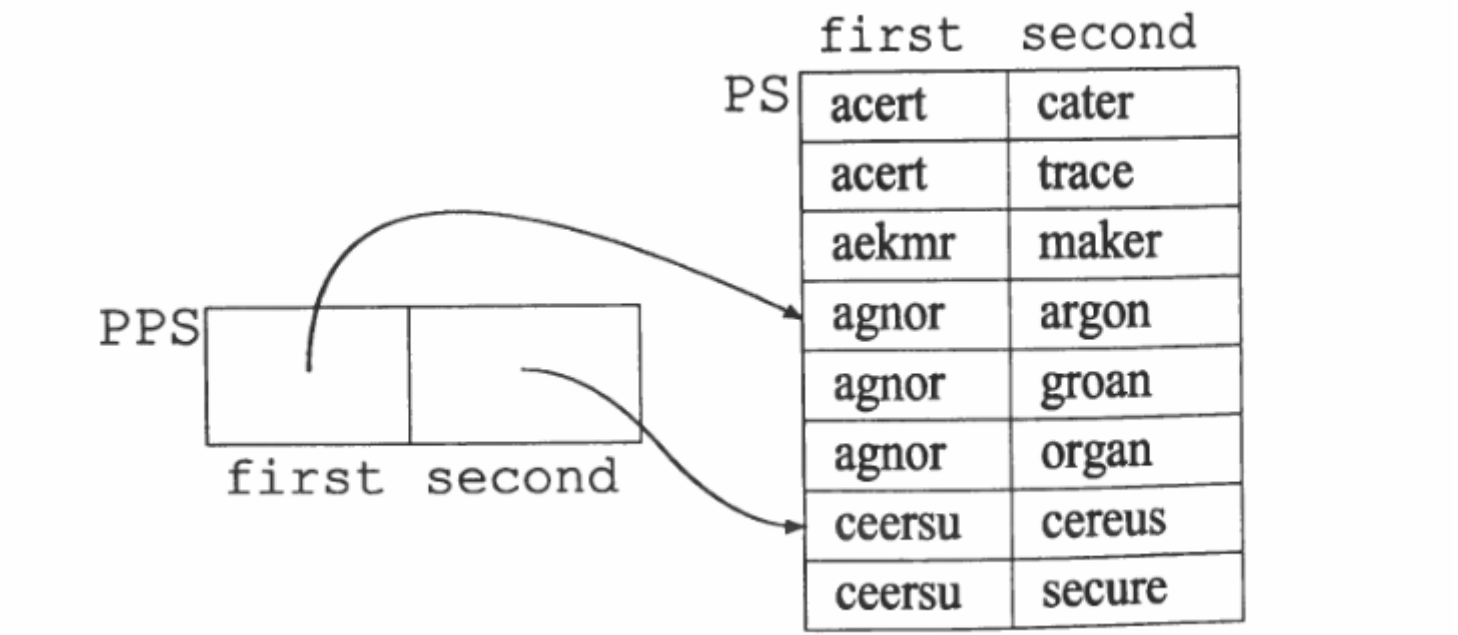
```
sort(word_pairs.begin(), word_pairs.end(), firstLess);
```

Использование предиката эквивалентности для поиска равных элементов

```
vector<PS>::const_iterator j = word_pairs.begin(),finis = word_pairs.end(), k;
        cout << "Output all group of anagram" << endl;
        while (true) {
                j = adjacent_find(j, finis, firstEqual);
                if (j == finis)
                        break;
                k = find_if(j + 1, finis, not1(bind1st(firstEqual, *j)));
                cout << " ";
                copy(j,k, ostream_iterator<string>(cout, " "));
                cout << endl;
                j = k;
        }
        return 0;
}
```

Группы анаграмм:
hasn't shan't
drawback backward
bacterial calibrate
cabaret abreact
bandpass passband
abroad aboard
banal nabla
balsa basal
saccade cascade
coagulate catalogue
ascertain sectarian
activate cavitate
vacuolate autoclave
caveat vacate
charisma archaism
maniac caiman
caviar variac
scalar lascar rascal sacral
causal casual
drainage gardenia
emanate manatee

Упорядоченные группы анаграмм по длине

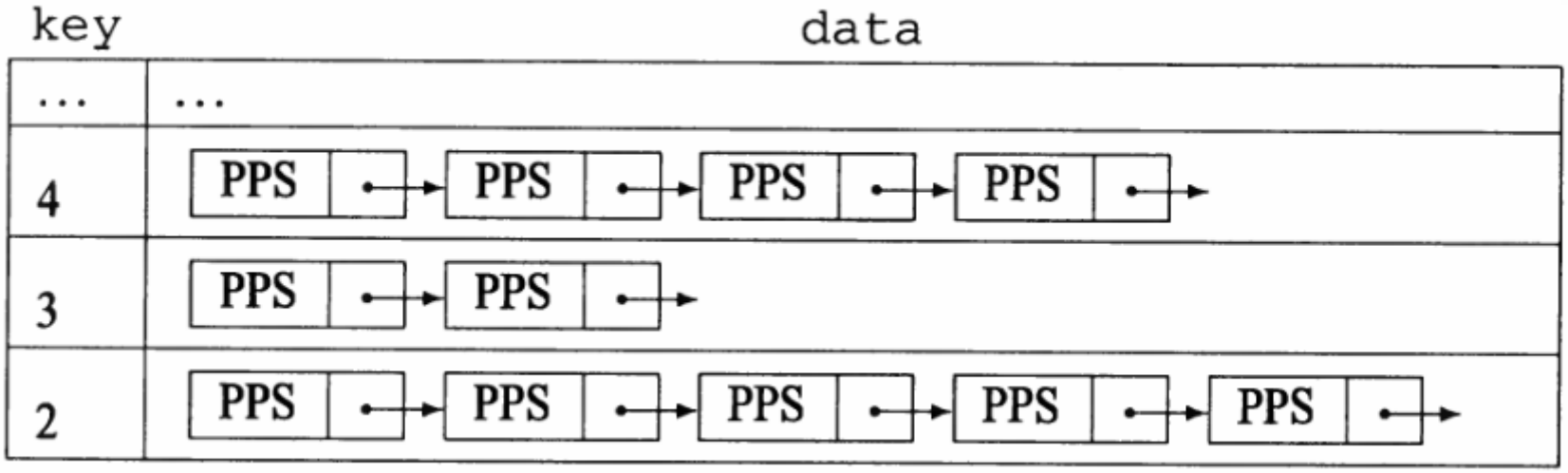


Определение структуры данных пар итераторов

```
typedef vector<PS>::const_iterator PSi;
typedef pair<PSi, PSi> PPS;
```

Хранение информации в отображении списков

```
// Отображение размеров групп на их списки
typedef map<int, list<PPS>, greater<int> > map_1;
  map_1 groups;
```



```
PSi j = word_pairs.begin(), finis = word_pairs.end(), k;
 while (true) {
   j = adjacent_find(j, finis, firstEqual);
   if (j == finis) break;
   k = find_if(j + 1, finis, not1(bind1st(firstEqual, *j)));
   if (k-j > 1)
     // Save the positions j and k delimiting the anagram
     // group in the list of groups of size k-j:
     groups[k-j].push_back(PPS(j,k));
   j = k;
 }
```

Вывод групп анаграмм в порядке убывания размеров

```
map_1::const_iterator m;
 for (m = groups.begin(); m != groups.end(); ++m) {
   cout << "\nAnagram groups of size " << m->first << ":\n";
   list<PPS>::const_iterator l;
   for (l = m->second.begin(); l != m->second.end(); ++l) {
     cout << "    ";
     j = l->first;  // beginning of the anagram group
     k = l->second; // end of the anagram group
     copy(j, k, ostream_iterator<string>(cout, " "));
     cout << endl;
   }
 }
```

Группы анаграмм:

Группы анаграмм размером 5:
crate carte cater caret trace

Группы анаграмм размером 4:
scalar lascar rascal sacral
bate beat beta abet
glare lager large regal
mantle mantel mental lament
peal pale leap plea
leapt plate pleat petal
slate steal stale least
mane name mean amen
mate team tame meat
pare pear reap rape
tea ate eat eta
cereus recuse rescue secure
edit tide tied diet
vile live evil veil
item mite emit time